KNOWLEDGE DRIVEN SYSTEM ARCHITECTURE TO SUPPORT COLLABORATIVE PRODUCT DEVELOPMENT IN THE EXTENDED ENTERPRISE

Karina Rodriguez Echavarria BSc

A thesis submitted in partial fulfilment of the
requirements of the University of Wolverhampton
for the degree of Doctor of Philosophy

June 2005

This work or any part thereof has not previously been presented in any form to the University or to any other body whether for the purposes of assessment, publication or for any other purpose (unless otherwise indicated). Save for any express acknowledgments, references and/or bibliographies cited in the work, I confirm that the intellectual content of the work is the result of my own efforts and of no other person.

Signature.....Karina.....

Date.....25th June 2005.....

Abstract

In recent times, the global engineering environment has led to the distribution of product life cycle information and knowledge affecting the collaboration throughout product development. Although information technologies, such as the Internet, provide a partial solution to support such collaboration, there is still a need to support decision making by providing the right information and knowledge in the place, time and format required by the geographically distributed companies. The sources of this knowledge are the experience of individuals, published literature, as well as the manufacturing process and resource capabilities. Hence, it ensures the production of a better and more cost effective product in less time.

The research presented in this thesis proposes a knowledge driven system architecture to support collaborative product development (KdCPD). Furthermore, a novel approach for identifying, capturing and representing knowledge of a geographically distributed extended enterprise was developed as part of the research. This knowledge representation, which is referred to as Manufacturing Knowledge Model, is the basis of the proposed system architecture.

In this research, a reference framework was adopted for the development of the KdCPD system architecture. This framework guided the identification of information and knowledge driven manufacturing activities as well as the modelling of the Manufacturing Knowledge Model. Based on this, a Knowledge driven Collaborative Product Development (KdCPD) system architecture was designed and a system prototype was implemented using object oriented enabling technologies. Finally, several experiments were conducted in the system prototype using several case studies in order to simulate the development of injection moulded parts among geographically distributed companies after the conceptual design has been agreed. The results of these experiments demonstrated how the KdCPD system supports decision making by providing the right information and knowledge in the place, time and format required. This confirmed the contribution of this research to the next generation of collaborative systems.

Acknowledgments

I would like to express my gratitude to everybody who in one way or another helped me throughout the course of this research.

First of all, I would like to express my gratitude to the University of Wolverhampton for granting me the scholarship that made this research possible. Thankfulness also goes out to the staff and IT technicians of the School of Engineering and Building Environment for their support. Special thanks to my second supervisors Dr. Chris Nwgaboso and Dr. Kevin Kibble for their advice regarding this thesis. I also want to thank Latmier Technologies, Arvin Meritor and Denso for providing information during this research as well as Excelon for providing the OODBMS Object Store™.

Very special thanks are due to Dr. Ahmed Al-Ashaab who provided me not only with his advice and supervision, but who also offered me his friendship and a family who always made me feel at home. I am indeed indebted to his family for all the enjoyable times we shared.

Finally, I would like to acknowledge my family in Mexico, my parents Antonio and Francisca, as well as my sisters Veronica, Deyanira and my brother Antonio. I am also very grateful to Laurens Vlaar, for his assistance in reviewing the text of this thesis, and for sharing with me the good and bad times of these years. There are no words to express my gratitude to all of them for their patience, support and encouragement throughout this endless journey in pursue of my dreams.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research rational

The manufacturing environment has expanded globally in recent years and competitiveness has intensified dramatically. This trend has been driven principally by world open market and growing customer expectations for products delivered quickly and at competitive prices. In this global environment, companies do not posses all the knowledge and resources they need but instead rely on buying them through contractual and cooperative partnerships with other companies, which are geographically distributed (Choo et al. 2000). This relationship among companies is called extended enterprise (Browne et al. 1997) and comprises the Original Equipment Manufacturers (OEM), its supply chain, subsidiaries, consultants, and partners affiliated with the life cycle of the product. As such, there is an increasing need to support the development of products in collaboration within the extended enterprise.

In addition, product development within the extended enterprise has led to the geographical distribution of the product life cycle information among the supply chain. This information includes both product data and enterprise knowledge. Research has been conducted in relation to capturing, structuring and sharing product data, such as geometrical data, bill of materials (BOM), testing, manufacturing and other engineering data. This data has been well represented through the ISO STEP 10303 (TC184/SC4 2004). However, enterprise knowledge is a relatively new area that has called the attention of both research and industrial communities. This is because knowledge is viewed as a company's asset, which has an impact on enhancing the productivity of the business. This knowledge usually resides within the memories of individuals, corporate information of past products and projects, literature and other sources (Caldwell et al. 2000). Moreover, it is related to the manufacturing process and resource capabilities of

the companies. It is critical to provide this knowledge in the right time, place and format to support engineering decision making throughout product development within the extended enterprise. Therefore, there is a need to provide not only a mechanism for distance communication, but also the knowledge and information required during collaborative product development (CPD).

Lately, various research initiatives have focused on developing Internet based applications to support collaborative design and manufacturing across the geographical barriers. In this thesis, this type of systems is referred to as Collaborative Product Development (CPD) system, which is defined as: "an Internet based computational architecture that supports the sharing and transferring of knowledge and information of the product life cycle amongst geographically distributed companies to aid taking right engineering decisions in a collaborative environment" (Rodriguez and Al-Ashaab 2002). Previous research initiatives have focused on sharing design geometric data in a collaborative environment during the design activity. Furthermore, the commercial software houses have come up with similar solutions, such as CoCreate (2003); Dassault Systemes (2003); Informative Graphics Corp. (2003); PTC (2003). However, the author believes there is still a need to support decision making throughout product development by providing product life cycle information and knowledge in the place, time and format required by the geographically distributed companies.

In order to provide knowledge to support product development, it is required to capture the knowledge of the extended enterprise. One type of such knowledge is manufacturing process information, which represents the process, resources and other constraints that limit the decisions that can be taken during the product life cycle. Previous research within the area of structuring manufacturing process information (Al-Ashaab 1994; Molina 1995; Young et al. 2001) has shown promising results for providing structured knowledge to engineers. Therefore, the author believes that a practical solution to the need of supporting CPD is to develop a knowledge web-based system architecture that relies on manufacturing process information. This information is referred to as manufacturing constraints in this thesis. In addition, the system architecture must support

a range of engineering activities that need to be performed in a collaborative manner. This work develops the research and commercial efforts further in order to provide a solution to support distance collaboration within the extended enterprise. The proposed system architecture will be referred to as KdCPD (Knowledge driven Collaborative Product Development) system architecture.

In order to demonstrate the research concept feasibility and its value, this work focuses on capturing knowledge related to process, resources and material constraints of the injection moulding process. The competitiveness in the injection moulding industry has made effective plastic product development very critical. For this reason, the production of quality and cost effective products in less time will enable this industry to remain competitive with stable economic growth. Hence, the injection moulding process is suitable to demonstrate the research concept.

1.2 Aim and objectives

The aim of this research is to develop a knowledge web-based system architecture for collaborative product development using manufacturing constraints to support the development of injection moulded plastic products.

In order to achieve this aim, the objectives of the research are stated as:

1. To research work related to best practices in CPD systems in order to identify the challenging research issues in this field.
2. To identify through a field study the industry's requirements and points of view related to the emerging technologies to support distance product development.
3. To define a methodology to support the development of a knowledge web based system architecture to support collaborative product development.
4. To identify, capture and represent the knowledge related to manufacturing constraints of the injection moulding process located among the geographically distributed partners within an extended enterprise.

5. To design a system architecture to support the new generation of knowledge driven collaborative injection moulded product development.
6. To develop and implement a functional prototype of the proposed knowledge driven collaborative injection moulded product development system.
7. *To validate the system through case studies.*

1.3 Research methodology

As shown in figure 1.1, the research followed a set of steps to propose and develop the Knowledge driven Collaborative Product Development (KdCPD) system architecture. The different activities of the research were conducted as follows:

1. An extensive review of research and commercial initiatives was performed in order to identify the technological requirements of the systems that support collaborative product development (see figure 1.1-a). The review, which is presented in chapter 2, also highlighted several research issues.

2. Parallel to the review, the industrial requirements were identified by performing a field study among three injection moulding companies within the UK (see figure 1.1-b). The field study and its results are described in detail in chapter 3.

3. Using both findings from the literature review and the field study, a set of research issues was identified. These issues, which are presented in chapter 4, sustain the author's hypothesis of the need of a knowledge based system architecture to support decision making during CPD. Chapter 4 also presents a methodology to develop a system architecture which addresses these research issues. This methodology is based on the CIMOSA (ESPRIT Consortium AMICE 1993) reference framework (see figure 1.1-c), which requires the modelling of product development activities and of other elements of an extended enterprise. The modelling techniques used as part of the reference framework are described in detail in the same chapter.

4. As shown in figure 1.1-d, the methodology followed to develop the system architecture guided the identification of information and knowledge driven manufacturing activities using IDEF0 (Colquhoun et al. 1993). The modelling of these activities is presented in chapter 5.

5. A Knowledge driven Collaborative Product Development (KdCPD) system architecture that addresses the research issues encountered during the literature review and the field study was then developed (see figure 1.1-f). Chapter 6 describes this architecture and its different components.

6. The activity modelling provided an initial insight of the required knowledge and UML (Object Management Group 2003) enabling technique was used to perform detail knowledge and information modelling. The development of these models is described in chapter 7 (see figure 1.1-e).

7. Thereafter, several case studies were produced in order to demonstrate how the models produced in the previous step could be used as a source of knowledge and information to support decision making during CPD. These are presented in detail in chapter 8.

8. A prototype of the proposed KdCPD system was designed and implemented using an object oriented technique (see figure 1.1-g). For this, technologies, such as Java, Java3D and the object oriented database manager Object Store™, were used as enabling implementation technologies. Chapter 9 and 10 describe the object oriented design and implementation of the KdCPD system prototype.

9. Finally, the implemented prototype was used for experimentation in order to demonstrate how the proposed KdCPD system architecture effectively supports collaborative product development. This experimentation is presented in chapter 11, followed by discussion of the experimental results in chapter 12. Conclusions and further work are discussed in chapter 13.

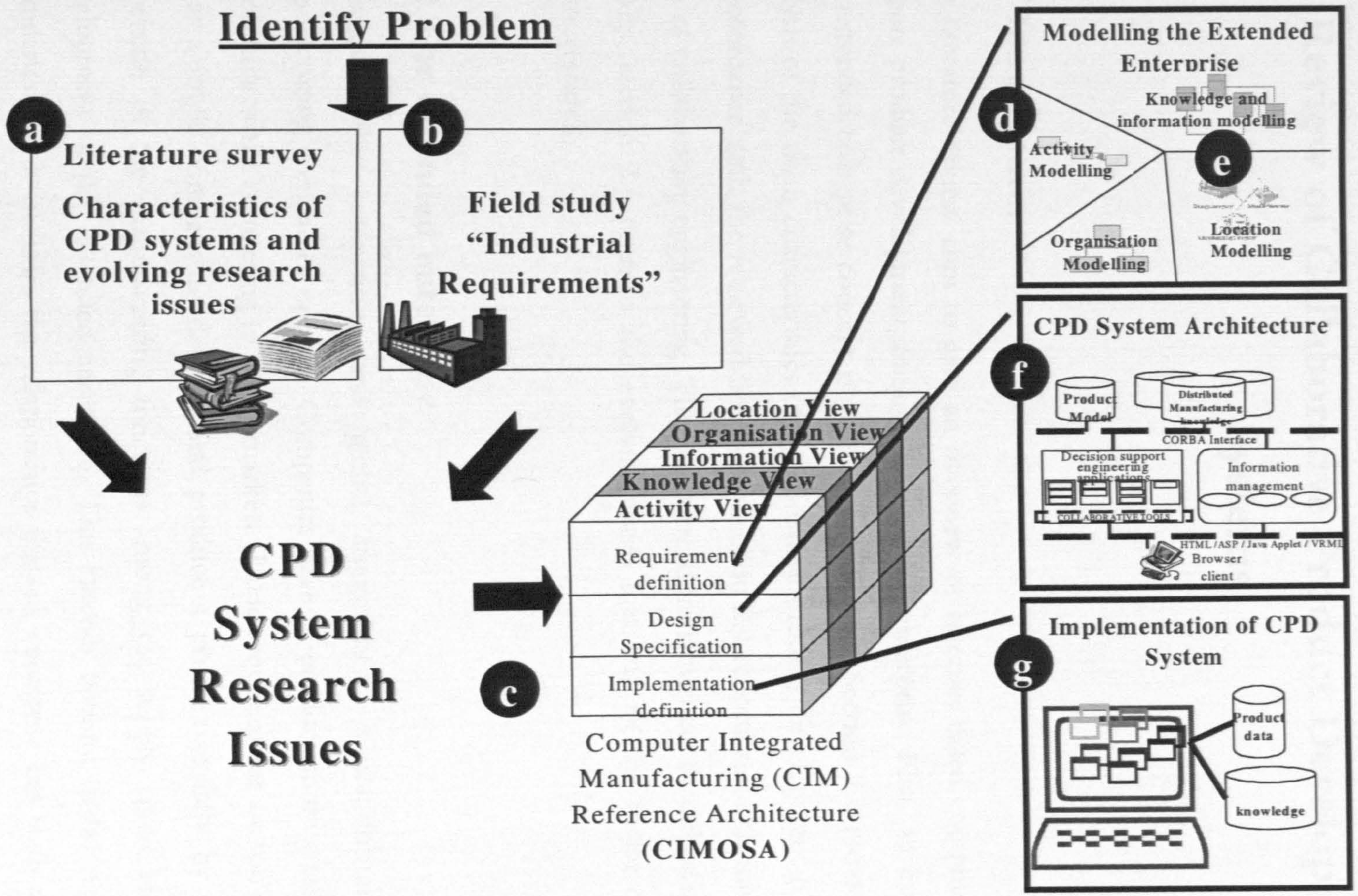


Figure 1.1 The methodology followed to develop a Knowledge driven Collaborative Product Development (KdCPD) system architecture

Chapter 2

Review of Collaborative Product Development Systems

2.1 Introduction

The literature review aims to give an overview of Internet based computer systems to support product development among the extended enterprise. First, an introduction to the extended enterprise concept is given in section 2.2. Section 2.3 provides a detailed analysis of the main characteristics of collaborative product development (CPD) systems in accordance with the reviewed literature. Section 2.4 reports research initiatives in the area of collaborative engineering. The commercial initiatives are described in section 2.5. Finally, section 2.6 presents the evolving issues that will be the major challenges for future research.

2.2 The extended enterprise

Nowadays, the competitive and global environment is transforming the way manufacturing companies operate. Companies have to produce more complex products in less time while competing in global markets. These requirements are too demanding to allow a single company to develop and produce a product entirely by its own. The dispersion of the manufacturing functions among the supply chain has led to the development of the extended enterprise. This extends beyond traditional organisation boundaries and it includes the relationships that an enterprise has with all its business partners.

The extended enterprise can be defined as a kind of 'enterprise', which is represented by all those organisations (or parts of organisations), customer, suppliers and subcontactors,

which are engaged collaboratively in the design, development, production and delivery of products to the end user (Browne, Hunt et al. 1997).

An extended enterprise model based on Browne et al. (1997) is used in order to describe and analyse this complex system further (see figure 2.1). This model represents a functional view of an extended enterprise and identifies the main building blocks for assembling an extended enterprise environment.

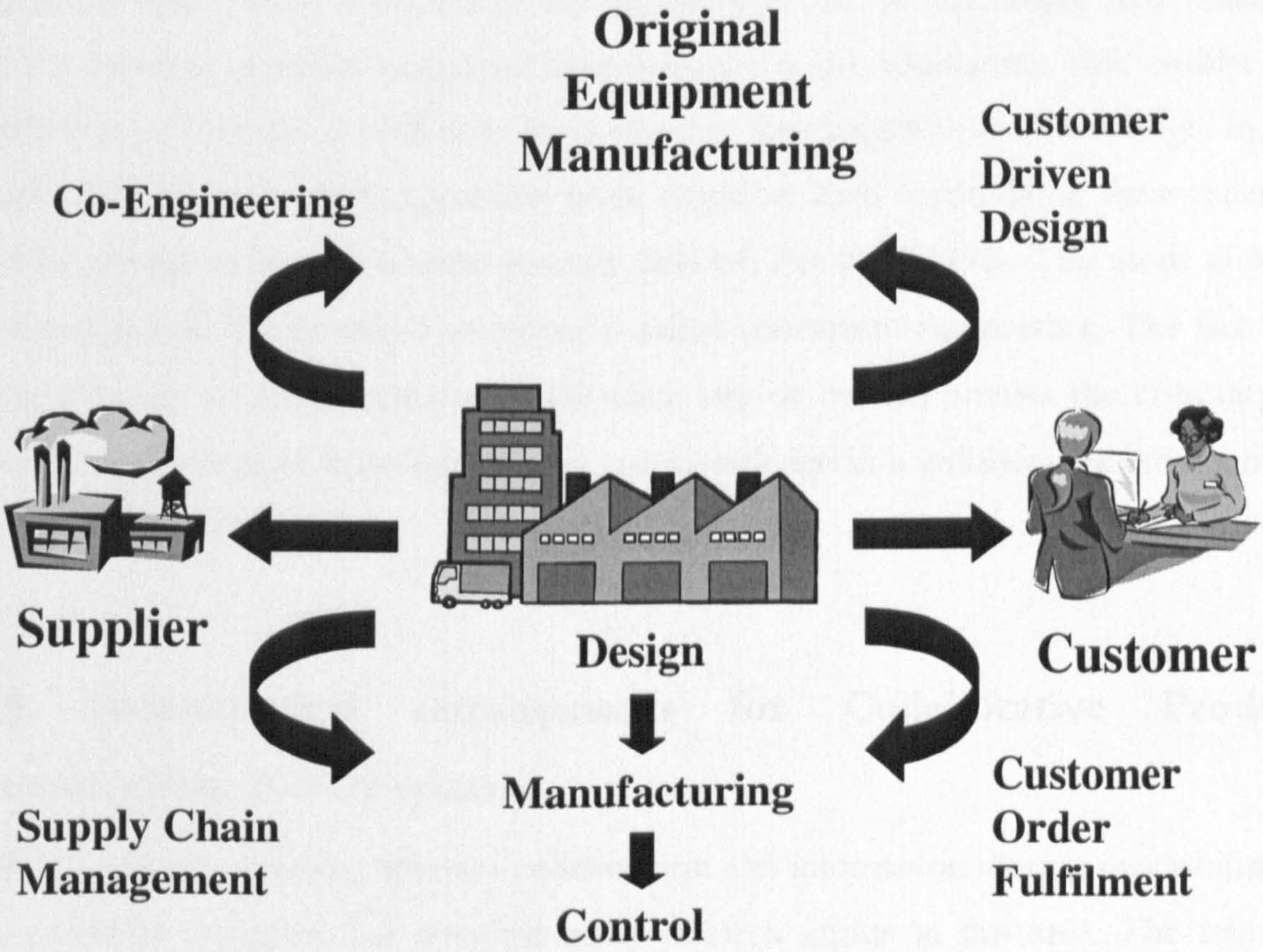


Figure 2.1 The extended enterprise model (Browne, Hunt et al. 1997)

The model is composed of a vertical and a horizontal axes. The vertical axis is composed of design, manufacturing and control issues, therefore representing the traditional, functional view of a manufacturing system. The vertical axis also represents the flow of information and material between the various functions and manufacturing. The horizontal axis represents the integration of the supplier, distributor and customer with

the manufacturing system in an extended enterprise. The horizontal axis also represents the flow of materials/products from the supplier, through to the customer.

Increasingly, the companies that belong to an extended enterprise are geographically distributed around the world. Hence, this distributed mode of working brings new challenges to the effective and efficient performance of the extended enterprise.

The success of CPD in the extended enterprise is greatly determined by the speed and efficiency with which information and knowledge can be exchanged and managed among partners. Another important success factor is the mechanism that enables the partners to collaborate, as well as to bring together their expertise and knowledge. In this working approach multiple specialists work together, each contributing their expertise and knowledge to achieve a better product (Sehdev, Fan et al. 1995). This mode of team working among the extended enterprise is called concurrent engineering. The fact that companies are no longer situated in the same city or country stresses the criticality of having mechanisms to share information and knowledge in a collaborative environment using accessible technologies.

2.3 Technological requirements for Collaborative Product Development (CPD) systems

The importance of having effective collaboration and information sharing mechanisms in the extended enterprise has attracted many research efforts in this area. The research community has, in the past few years, developed several research initiatives to propose CPD systems (Roy, Bharadwaj et al. 1997; Gupta, Paredis et al. 1998; Kim, Kim et al. 1998; Biennier and Favrel 1999; Chang, Lu et al. 1999; Törlind 1999; Abrahamson, Wallace et al. 2000; Caldwell, Clarkson et al. 2000; Domazet, Yan et al. 2000; Lu and Cai 2000; Rezayat 2000; Sevy, Zaychik et al. 2000; Zhuang, Chen et al. 2000; Anderson and Abdalla 2001; Huang and Mak 2001; Jae, Hyun et al. 2001; Li, Bracewell et al. 2001; Qiang, Zhang et al. 2001; Chung and Kunwoo 2002; Su, Ji et al. 2002; Qin, Harrison et al. 2003).

The literature review has highlighted several technological requirements that must be addressed in order to develop adequate technologies for this type of systems. These are:

1 Information system architecture: Both information models and engineering applications are integrated within a framework in a structured and transparent manner using communication protocols between the elements of the system (Molina, Al-Ashaab et al. 1995).
2 Communication tools: Tools to enable the visual/audio communication amongst geographically distributed team members.
3 Virtual team management tools: Software applications to coordinate and administrate the distributed team members.
4 Product Model: A software representation of form and data that describes a product throughout its life cycle (Young and Bell 1992).
5 Engineering applications: Software to support engineering decision making throughout product development.
6 Product geometric representation: Software applications that facilitates the visualisation of product design amongst the geographically distributed team members.
7 Integration with CAD/CAM/CAE commercial software: Interface software to import/export files from commercial CAD/CAM/CAE systems.
8 Knowledge representation: Documentation of learning lessons and other generic rules, which are stored in a repository of information.
9 Project management tools: Software applications to coordinate and administrate product development activities and their required resources over the Internet.

Table 2.1 exhibits the reviewed CPD systems illustrating the technological requirements they support. The following subsections present in more detail some of the key technological requirements.

Table 2.1 Technological requirements currently addressed by the reviewed CPD system

CPD Systems \ Technological Requirements	Information system architecture	Communication tools	Virtual Team Management	Product Model	Engineering applications	Product geometric representation	Integration with CAD/ CAM/ CAE	Knowledge representation	Project Management
DOME by Abrahamson *et al.*2000	★				★		★		
DISCS by Anderson *et al.* 2001	★	★		★	★	★	★		
Biennier *et al.* 1999	★				★				
WebCADET by Caldwell *et al.* 2000	★				★			★	
Chang *et al.* 1999	★	★		★	★		★	★	
Chung *et al.* 2002	★				★		★		
SOMF by Domazet *et al.* 2000	★		★	★	★	★			★
CODES by Gupta *et al.* 1998	★		★	★	★	★	★		
Design for X by Huang *et al.* 2000	★			★	★			★	
NetFEATURE by Jae *et al.* 2001	★			★	★	★			
CyberView by Kim *et al.* 1999		★		★	★	★	★		
EDSE by Li *et al.*2001					★			★	
STARS by Lu *et al.*2000	★		★		★				★
WPDSS by Qiang *et al.* 2001	★	★			★	★	★		
Qin *et al.* 2003					★	★	★		
Enterprise-Web by Rezayat 2000	★	★	★		★	★			
Roy *et al.* 1999		★		★	★	★	★	★	
Collab. Studio by Sevy *et al.* 2000	★	★			★	★	★		
Su D. *et al.* 2002	★	★			★	★	★		
DCEE by Törlind 1999		★			★	★	★		
CyberEye by Zhuang *et al.* 2000	★	★	★		★	★			

2.3.1 Information system architecture

Considerable research has been undertaken to establish architectures for developing information systems. These architectures define the integration environment and the communication protocols between the system elements, such as information models and engineering applications, in a structured and transparent manner. Examples of architectures are: National Industrial Information Infrastructure Protocols (National Industrial Information Infrastructure Protocols 2001); RM-ODP (International Comittee for Information Technology Standards 2003); Distributed Computing Environment (The Open Group 2000) and CORBA (Object Management Group 2003).

The review of the CPD systems has highlighted that most of the initiatives that define a structured architecture use CORBA as a reference (Törlind 1999; Abrahamson, Wallace et al. 2000; Domazet, Yan et al. 2000; Rezayat 2000). CORBA (Common Object Request Broker Architecture) is a reference architecture used to implement computer systems that operate in a distributed environment and communicate over networks (Object Management Group 2003).

2.3.2 Communication tools

In order to support the interaction between geographically distributed team members the reviewed systems utilised synchronous and asynchronous communication tools. Synchronous tools are used for real time communications, such as video and audio conferencing, whiteboard, chat sessions, as well as sharing geometric models or CAD files. In other words, these tools provide a virtual meeting environment. Asynchronous tools are used in non real time communications, i.e., email or file transfer applications.

The inclusion of communication tools has been well addressed by most of the reviewed CPD systems. Roy et al. (1997) and DCEE (Törlind 1999) provide a video conference tool, while Collaborative Studio (Sevy, Zaychik et al. 2000) and Su et al. (2002) provide an audio conference tool. Other systems provide a whiteboard tool (Roy, Bharadwaj et al. 1997; Chang, Lu et al. 1999; Sevy, Zaychik et al. 2000; Su, Ji et al. 2002). Chang et al. (1999), CyberView (Kim, Kim et al. 1998), Su et al. (2002), DCEE (Törlind 1999) and CyberEye (Zhuang, Chen et al. 2000) provide an environment to visualise the product geometry in real time.

2.3.3 Virtual team management

Some researchers agree it is necessary to administrate the virtual team members in order to support CPD effectively (Gupta, Paredis et al. 1998; Domazet, Yan et al. 2000; Lu and Cai 2000; Rezayat 2000; Zhuang, Chen et al. 2000). This management enables the provision of information to the appropriate team member.

Examples of virtual team management application are provided by Gupta et al. (1998), Rezayat (2000) and Zhuang et al. (2000). In these systems, the information of the team members, such as name, role in the project and email, is stored in a database. According to the role the engineer plays in the product development, the systems give different information access rights to each member of the team. The information is also used to schedule collaborative sessions between engineers by sending email notifications to the required engineers in a particular session.

2.3.4 Product model

The product data is used and produced by different engineering applications throughout the product development process. This data is usually stored in what is called a Product Model. This model constitutes an important element of any CPD system and its content and structure relates to the supported engineering applications. As presented in section 2.3.5, most of the reviewed CPD systems are concerned with the design activity. For this reason, most of the Product Models have been structured to capture product design data, mainly geometric data and BOM.

The Product Models included in the reviewed systems have been structured and implemented in either of the following forms:

- Based on the ISO standard STEP 10303 AP-203, which is related to geometric data of the product (Kim, Kim et al. 1998; Törlind 1999; Domazet, Yan et al. 2000; Anderson and Abdalla 2001).
- Based on a non standard structure and implemented with commercial databases. Chang et al. (1999), Jae et al. (2001), Gupta et al. (1998), Qin et al. (2003) and Roy et al. (1997) developed their own product design and manufacturing data representations. The information captured in these models includes product features (Roy, Bharadwaj et al. 1997; Chang, Lu et al. 1999; Jae, Hyun et al. 2001), geometric data (Roy, Bharadwaj et al. 1997; Gupta, Paredis et al. 1998; Chang, Lu et al. 1999) and machine tools (Roy, Bharadwaj et al. 1997).

2.3.5 Engineering applications

Effective CPD could be achieved by using applications that support engineering decision making. Therefore, engineering applications should be included in any CPD system. Furthermore, it is critical that some of them are performed in a collaborative environment. For example, the product design, which includes conceptual design, design analysis, prototyping and optimisation, requires the interaction of several team members to produce better results. For this reason, the real time communication tools, presented in section 2.3.2, play an important role in supporting the interaction between the multidisciplinary team.

Table 2.2 Product life cycle activities supported by the reviewed CPD systems

CPD Systems	Product life cycle activities
DOME by Abrahamson *et al.*2000	Conceptual design
DISCS by Anderson *et al.* 2001	Conceptual design
Biennier *et al.* 1999	Conceptual design
WebCADET by Caldwell *et al.* 2000	Conceptual design
Chang *et al.* 1999	Conceptual design, design for manufacturability
Chung *et al.* 2002	Conceptual design
SOMF by Domazet *et al.* 2000	Conceptual design
CODES by Gupta *et al.* 1998	Conceptual design, design for X, manufacturing process plan.
Design for X by Huang *et al.* 2000	Conceptual design, design for X, manufacturing process plan
NetFEATURE by Jae *et al.* 2001	Conceptual design
CyberView by Kim *et al.* 1999	Conceptual design
EDSE by Li *et al.*2001	Conceptual Design
STARS by Lu *et al.*2000	Conceptual design
WPDSS by Qiang *et al.* 2001	Conceptual design
Qin *et al.* 2003	Conceptual design
Enterprise-Web by Rezayat 2000	Conceptual design
Roy *et al.* 1999	Conceptual design, design for X, prototyping, manufacturing process planning
Collab. Studio by Sevy *et al.* 2000	Conceptual design
Su D. *et al.* 2002	Design specifications, conceptual design, manufacturing process planning
DCEE by Törlind 1999	Conceptual design
CyberEye by Zhuang *et al.* 2000	Conceptual design

Table 2.2 illustrates different product life cycle activities supported by the reviewed systems. As shown in the table, most of the research efforts have been directed to support the design activity. The following subsections will review the key activities of the

product life cycle that the systems are supporting, as well as the engineering applications that the systems provide.

2.3.5.1 Conceptual design

This activity involves collaboration and, consequently, extensive communication among the team members in order to create, analyse and evaluate design alternatives. The systems reviewed have supported the following three approaches:

a. Common access to design data: In this approach, the collaboration is achieved by sharing product data (see figure 2.2-a). There is no real time collaborative visualisation of the product designed. The data is downloaded from a database, or from a commercial Product Data Management (PDM) system. The emphasis of this approach is on sharing product data, mainly design data, such as product specifications, geometry and BOM (Gupta, Paredis et al. 1998; Biennier and Favrel 1999; Törlind 1999; Abrahamson, Wallace et al. 2000; Domazet, Yan et al. 2000; Lu and Cai 2000; Rezayat 2000; Zhuang, Chen et al. 2000; Anderson and Abdalla 2001; Li, Bracewell et al. 2001; Chung and Kunwoo 2002; Qin, Harrison et al. 2003).
b. Collaborative visualisation of the product: As shown in figure 2.2-b, this approach allows the engineers to visualise the 3D model of a product previously designed. This visualisation is done in real time and does not allow the modification of the 3D model in real time (Roy, Bharadwaj et al. 1997; Kim, Kim et al. 1998; Chang, Lu et al. 1999; Törlind 1999; Sevy, Zaychik et al. 2000). In order to support further the collaboration between the team members, some systems also provide synchronous communication tools (i.e. chat sessions, videoconferencing and whiteboard).
c. Collaborative design of the product: This approach allows the geographically distributed designers to visualise and modify the product geometry in real time (see figure 2.2-c). Qiang et al. (2001) and Su et al. (2002) proposed a system where the designers collaborate when designing a product geometry by concurrently using a commercial CAD system. By using this approach any user is able to modify the geometry, while other engineers are able to see the changes in real time. Other

commercially available initiatives, known as collaborative product commerce systems, use a similar approach, i.e. Alibre (Alibre Inc. 2003).

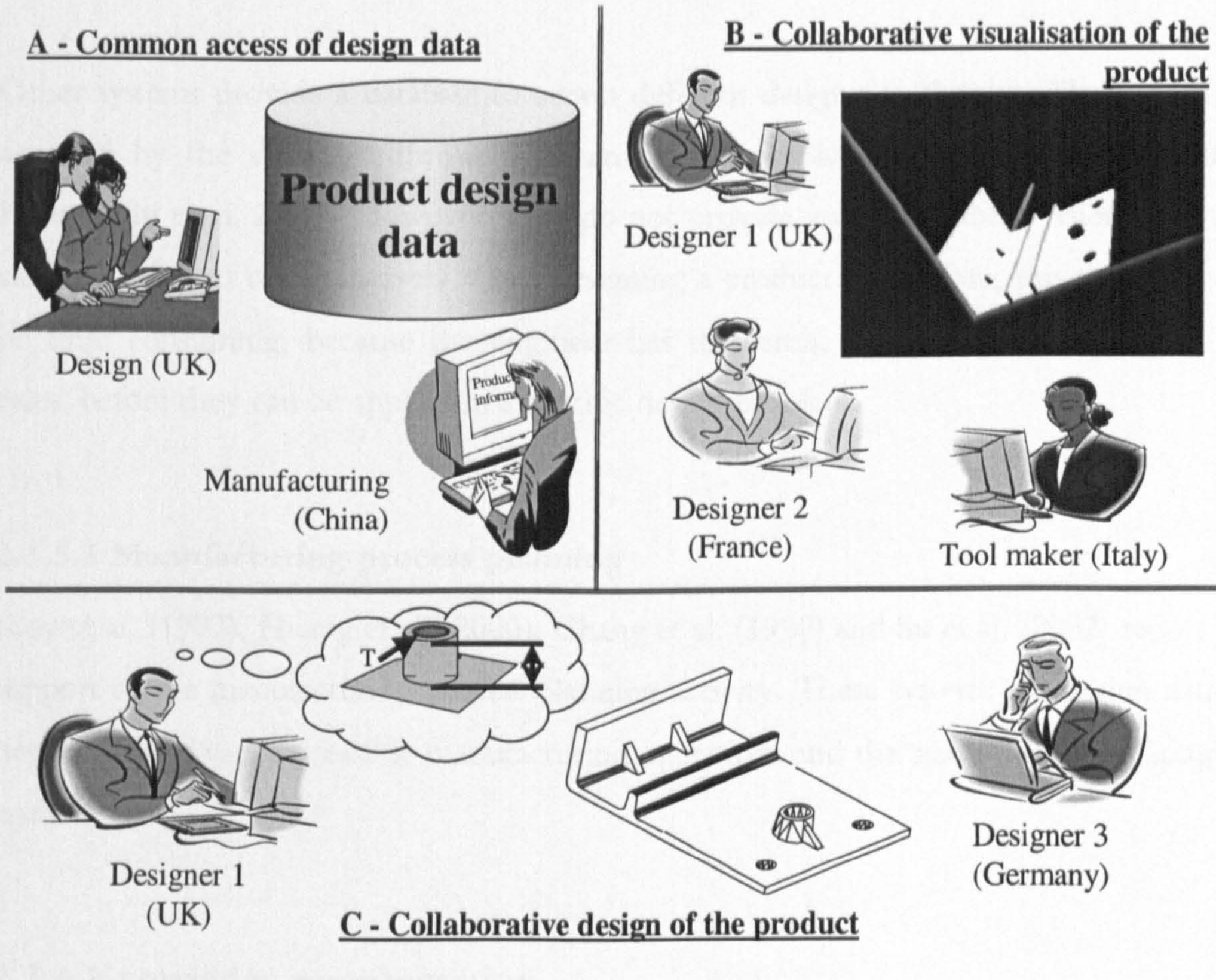


Figure 2.2 Different approaches to support collaboration during conceptual design

2.3.5.2 Design for X

Design for X is an approach to consider various issues involved in the design of a product, such as manufacturing, assembly, reliability or service. In order to evaluate these issues, it is necessary to capture design for X rules, which normally reside in books or in the head of the team members.

Chang et al. (1999) provide a design for manufacturing application, where the evaluation of the manufacturability is based on geometric constraints of the features. Gupta et al. (1998) provide a design for assembly application, where the product is evaluated to avoid

physical overlaps. Furthermore, Roy (1997) provides a casting analysis service. These systems provide feedback about design problems but they do not offer advice on how to correct these problems.

Other systems provide a database to access different design for X rules. These rules are accessed by the designers through a search engine (Caldwell, Clarkson et al. 2000; Huang, Shi et al. 2000). However, they do not provide an environment where the rules can be evaluated collaboratively when designing a product. Therefore, this approach can be time consuming, because the engineer has to search, understand and interpret the rules, before they can be applied in a specific design problem.

2.3.5.3 Manufacturing process planning

Roy et al. (1997), Huang et al. (2000), Chang et al. (1999) and Su et al. (2002) report the support of the manufacturing process planning activity. These systems use design data to determine the various feasible manufacturing operations and the associated manufacturing costs.

2.3.6 Knowledge representation

Knowledge related to product development should be captured in order to support decision making. There are different opinions as to what this knowledge includes. For this reason, the author has classified knowledge in the following types:

1. Product data: some authors consider as knowledge the documents or files related to a product. Examples of this type of data are product specifications, CAD files, design analysis and market studies (Höfling 1999). This data is considered useful when decisions need to be taken during product development. However, if the data is incorrect, this will be passed on throughout the product life cycle potentially causing problems at later stages.
2. Previous case history: the data about previous projects and the rationale for making decisions are also considered to be useful during current projects. The product data generated during previous projects is captured in an information repository

(Rupprecht et al. 2000, Gruber 1992, Bramall et al. 2001, Li et al. 1998, Zaychik et al. 2000). This approach could be useful, but is also time consuming as the relevant information has to be found, understood and applied. It also has the same disadvantage as the previous type of knowledge because the incorrect data can be reproduced if this was stored without noticing its inaccuracy.

3. Manufacturing constraints: the engineering decisions, related to the various stages of product development, are limited by various considerations, such as technological, process, resource, material cost or environmental considerations. For example, in order to design an injection moulded product there are certain characteristics of the process that need to be considered, such as the capability of producing only thin walled products. The polymer material may constrain the maximum and minimum wall thickness. The resources available in the company may constrain where the product will be produced. This knowledge is available most of the time from the experience of the engineers, in books or other documents.

Some attempts have been made to capture this type of knowledge in the form of ontologies or artificial intelligent rules to support isolated applications (Roy, Bharadwaj et al. 1997; Chang, Lu et al. 1999; Rodgers, Caldwell et al. 2001; Shi, Huang et al. 2001). One of the approaches is to store these constraints in a database. These systems only provide the capability for engineers to access and review the constraints rather than applying those on real product data to support decision making.

2.4 Collaborative engineering research initiatives

Increasingly, the research community is embarking on research initiatives to support the collaboration of engineers during the design of complex systems, such as electronic systems. Some examples of these initiatives are Bauer et al. (2001), Fliedner, Lee et al. (2003), Shyamsundar and Gadh (2001), Wang et al. (2003) and Zhan et al. (2003), which have focused in supporting the workflow coordination and the information and software sharing amongst the geographical distributed engineers. These initiatives main purpose is

in sharing project data and providing communication tools rather than in sharing knowledge to support decision making.

2.5 Commercial initiatives

The commercial initiatives to support the interactions among the extended enterprise have increased in the past few years. For the development of this software, many vendors rely on other commercial systems, such as Enterprise Resource Planning (ERP) or Product Data Management (PDM) systems. Web interfaces and communication tools have been added to these systems and they are now referred to as either Collaborative Product Development or Collaborative Product Commerce (CPC) solutions.

In this review, the commercial CPD systems have been classified into two types according to their functionality: collaborative design systems and data sharing systems. The following subsections will describe both types of systems in more detail.

2.5.1 Collaborative design systems

These types of systems have been described in section 2.3.5.1. Their main functionality is to provide an environment where the geographically distributed designers can visualise and modify the product geometry in real time. The designers are able to work collaboratively on the same geometry, by either of the following approaches:

- By enabling any designer to modify the solid geometric model in real time during the collaborative session (Alibre Inc. 2003).
- By enabling only one designer to modify the solid geometric model in real time while the others are just enabled to visualise the changes (CoCreate 2003; Dassault Systemes 2003; Informative Graphics Corp. 2003; PTC 2003).

Regardless of the method of collaboration used, all of these systems provide communication tools for the interaction of the geographically distributed designers.

In addition, other applications, such as project management (CoCreate 2003; Dassault Systemes 2003) and product data repositories (Alibre Inc. 2003; CoCreate 2003; Dassault Systemes 2003) are also provided.

2.5.2 Data sharing systems

The main functionality of these systems is to provide shared repositories of data where the team members can access project information despite their geographical location. Most of these commercial systems were already sold as Product Data Management systems, adding web interfaces and communication tools to their existing software (Agile Software 2003; Alventive Inc 2003; Centric Software 2003; Dassault Systemes 2003; e2open 2003; EDS 2003; Matrix One 2003; Oracle 2003; PTC 2003; TDCI Solutions 2003; Webscope Inc 2003). In addition, software that have CAD heritage are particularly strong in the visualisation of the product geometry (Dassault Systemes 2003; EDS 2003; PTC 2003).

Other functionalities offered by these systems are project management (Agile Software 2003; Alventive Inc 2003; Centric Software 2003; Dassault Systemes 2003; e2open 2003; Integrated Development Enterprise 2003; Matrix One 2003; Oracle 2003; Primavera Systems Inc 2003; Schlumberger Limited 2003; TDCI Solutions 2003) and virtual team management (Agile Software 2003; Alventive Inc 2003; Centric Software 2003; Dassault Systemes 2003; EDS 2003; Eurostep Commercial Solutions AB 2003; Integrated Development Enterprise 2003; Matrix One 2003; Oracle 2003; Primavera Systems Inc 2003; PTC 2003; Schlumberger Limited 2003; TDCI Solutions 2003).

2.6 Evolving research issues highlighted during the literature review

The analysis of the CPD systems clearly illustrated that each research group or software company has emphasised on one or two technological requirements. Almost all the CPD systems have focused on supporting the design activity. Progress has also been achieved in visualising and sharing product geometry. However, a complete solution that addresses all

the requirements has not yet been achieved. The following observations have been highlighted by the review as the research issues for the next generation of CPD systems:

- Several of the proposed CPD systems are storing knowledge but are not providing decision making support in a collaborative environment.
- A large number of CPD research and commercial initiatives have been directed towards the development of databases to store product data files, mainly design data, or product design history. These databases only work as information repositories during the design activity providing search engines to access the information. They do not use this information to support the engineers when making decisions.
- Current CPD systems do not support all the key activities of the product life cycle. Most of them only support the design activity, in particular the conceptual design activity.
- The reviewed CPD systems do not have structured product data of all the key activities of the product life cycle. The systems focus on handling data about the design, mainly the geometry. Only few systems are using standards like STEP to structure this data, but even these systems are using the geometric part of the standard.

Chapter 3

Industrial Requirements of Collaborative Product Development

3.1 Introduction

Following the literature review, a field study was conducted in order to investigate the industrial requirements for a CPD system. The methodology is described in detail in section 3.2, followed by the results of the field study in section 3.3. Finally, a set of research issues is presented in section 3.4.

3.2 Field study methodology

The objectives of the field study were the following:

- To investigate the industrial need to collaborate with the customer, supply chain and other partners.
- To understand the current mechanism of communication between the companies and their supply chain when such collaboration exists.
- To identify the best mechanisms to achieve effective collaboration according to the industrial needs.

A quantitative research method, by means of the questionnaire, was used to achieve these objectives. This method was selected as the questionnaire was found to be more suitable to numerically measure the responses. Its design was based on the information collected during the literature review.

The questions were arranged into four groups, namely:

1. The collaboration need with the supply chain.

2. The current and preferred communication mechanisms.
3. The information and knowledge that need to be shared with the supply chain.
4. Cultural issues that may have an impact on distance collaboration.

The questionnaire was designed using a closed approach, which means that a predefined set of answers was given for each question. Table 3.1 presents the first group of questions as an example. This group is further described in the following paragraphs in order to explain the design of the questionnaire in more detail. The complete questionnaire is included in appendix A.

Table 3.1 Example of questions related to the collaboration need with the supply chain

How important is it for your company to collaborate with partners and supply chains that are geographically distributed? ☐ Not important ☐ Regular ☐ Important ☐ Crucial
How important is it to share the knowledge (how to do things) with other engineers involved in product development within the same company and the supply chain? ☐ Not important ☐ Regular ☐ Important ☐ Crucial
How important is it to share the product data with other engineers involved in product development within the same company and the supply chain? ☐ Not important ☐ Regular ☐ Important ☐ Crucial
Do you think a computer software that helps you to collaborate and share knowledge and product information through Internet with the people situated in another place will help you to improve your work? ☐ Yes ☐ No

The questions presented in table 3.1 aimed to determine the engineer's perception of product development in collaboration with the supply chain and other partners. As shown in table 3.1, the answers provided for the first three questions represent a scale of relative importance for each referred issue. For example, in the question *"How important is it for your company to collaborate with partners and supply chain that are geographically distributed?"*, the answers range from "Not important", which means that this issue is not even considered, to "Crucial" which means that this issue is extremely important. The

last question in table 3.1 depicts the engineer's perception regarding the use of a computer software to support them to collaborate as well as to share knowledge and product information during product development.

3.2.1 Sample of the field study

The field study sample included 20 engineers of three injection moulding manufacturing companies in the UK. The number of engineers and of companies proved to be sufficient to gather a well balanced opinion of the industry. This is because the companies comprise a good sample of medium to large organisations that are part of global extended enterprises. In addition, these companies are involved in different aspects of plastic injection moulding, such as product design, mould design and fabrication, as well as the processing of the plastic parts. Each of these companies is described below:

- Company No. 1 (Medium): is a supplier of plastic injection moulding systems for the automotive industry. The company owns facilities in North America, UK and mainland Europe and supports its partners with design and manufacturing capabilities.
- Company No. 2 (Medium): is a supplier of access control systems for the automotive industry. It is based in United States and their plants are located within 3 continents: America, Europe and Asia. Its capability includes both design and manufacturing.
- Company No. 3 (Large): is the world's largest supplier of air conditioning systems and components for automobiles. It has close ties with the biggest OEMs all over the world. Its core technology is developed in Japan and it owns facilities in America, Europe and Asia.

The sample selected from each company was composed of employees involved in different activities of the product development. As such, a 25% were product engineers, 25% manufacturers, 15% toolmakers, 15% from the sales and marketing department, 10% from prototyping and testing and 10% project managers.

3.2.2 Limitations of the field study

It is important to note that the results of the field study may be affected by the fact that the engineers only have knowledge of technologies they have previously used. Therefore, they may not be aware of the advantages or disadvantages of new technologies. However, the results are still valid because the intention is to identify the industrial requirements for effective CPD based on current problems engineers are faced with. Other results may be influenced by the type of work the engineers do. For example, the designers may give more importance to product specifications than engineers working in the manufacturing department. For this reason, a selection of engineers from different departments has been interviewed.

3.3 Findings of the field study

In the following subsections the results of the four groups of questions of the field study are discussed.

3.3.1 Results from group 1: the collaboration need with the supply chain

One of the main findings of the field study is that the distance collaboration among the extended enterprises is crucial due to the companies' involvement in international manufacturing alliances. This result is illustrated in figure 3.1, where nearly 100% of the engineers considered the collaboration either important or crucial in the current product development practice and the sharing of knowledge and product data.

In addition, all the engineers believe that their work could be improved through the use of a computer software which supports their collaboration as well as the sharing of knowledge and product information (see figure 3.2). This result confirms the importance of the development of a CPD system to further support the extended enterprise.

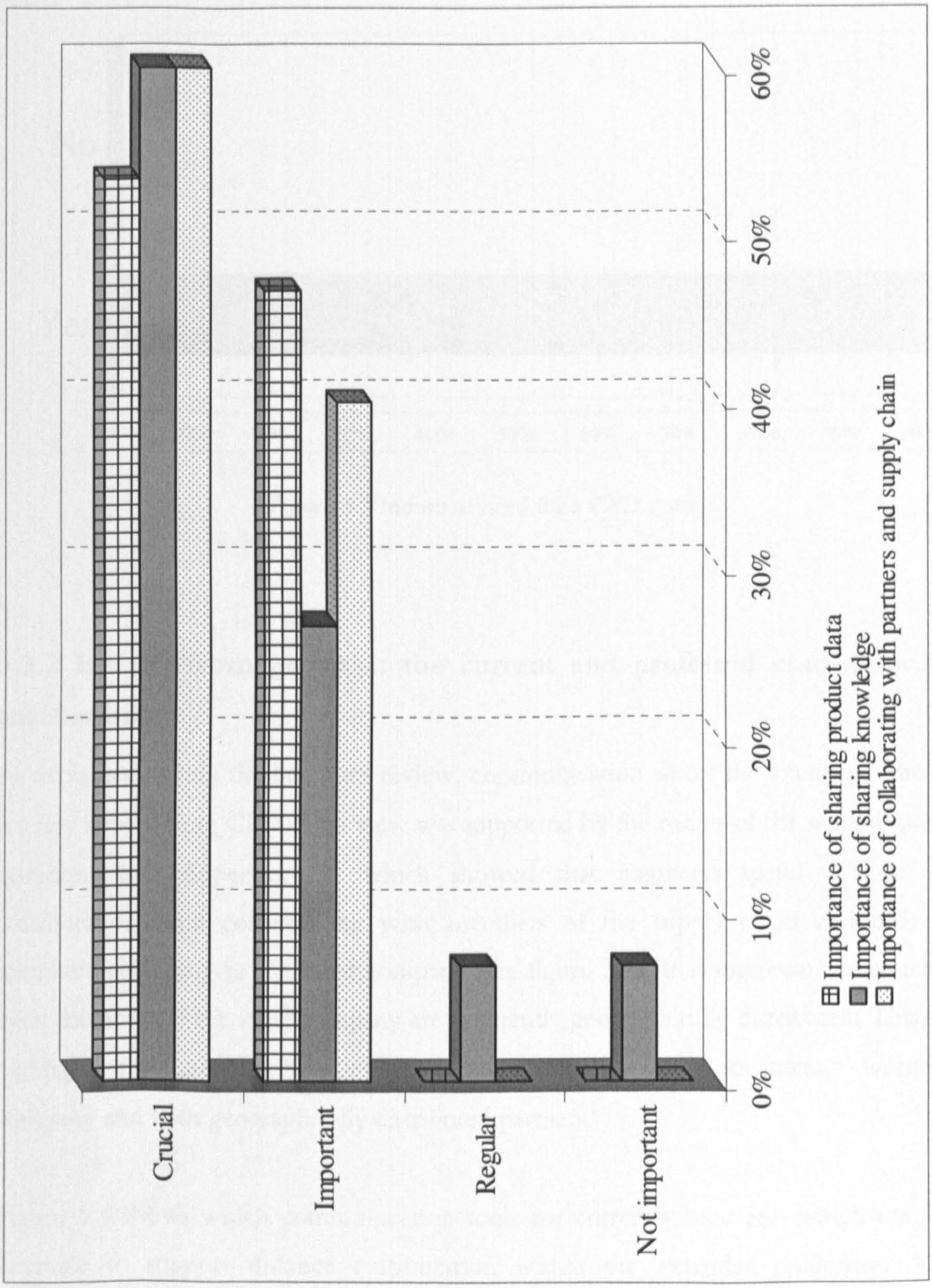


Figure 3.1 Importance of collaborating and sharing information and knowledge with the extended enterprise

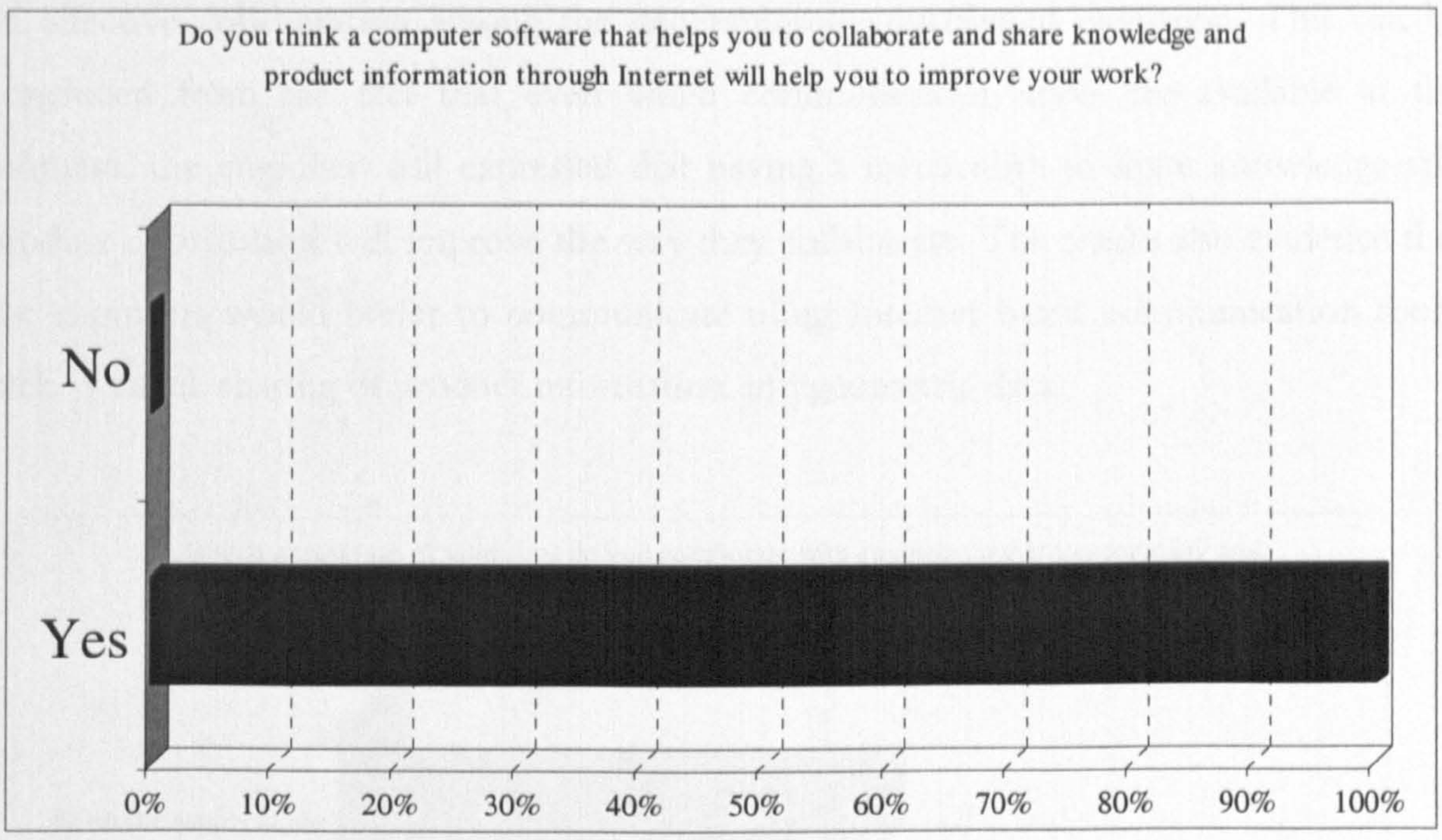


Figure 3.2 Industrial need for a CPD system

3.3.2 Results from group 2: the current and preferred communication mechanisms

As explained during the literature review, communication across the extended enterprise is a key issue during CPD. This view was supported by the results of the second group of questions (see Appendix A), which showed that engineers spend 37% of their collaboration time collaborating with members of the supply chain and 63% with members belonging to the same company (see figure 3.3). It is important to notice that even members of the same company are frequently geographically distributed. This result highlights the need to have effective communication tools to interact within the company and with geographically distributed partners.

Figure 3.4 shows which communication tools are currently used and which are found desirable to support distance collaboration within the extended enterprise. At the moment, communication tools such as phone and email are the most popular. However, it is important to highlight that having communication tools available does not guarantee

an effective collaboration among the geographically distributed engineers. This can be concluded from the fact that even when communication tools are available at the moment, the engineers still expressed that having a mechanism to share knowledge and product information will improve the way they collaborate. The results also evidence that the engineers would prefer to communicate using Internet based communication tools, such as email, sharing of product information and geometric data.

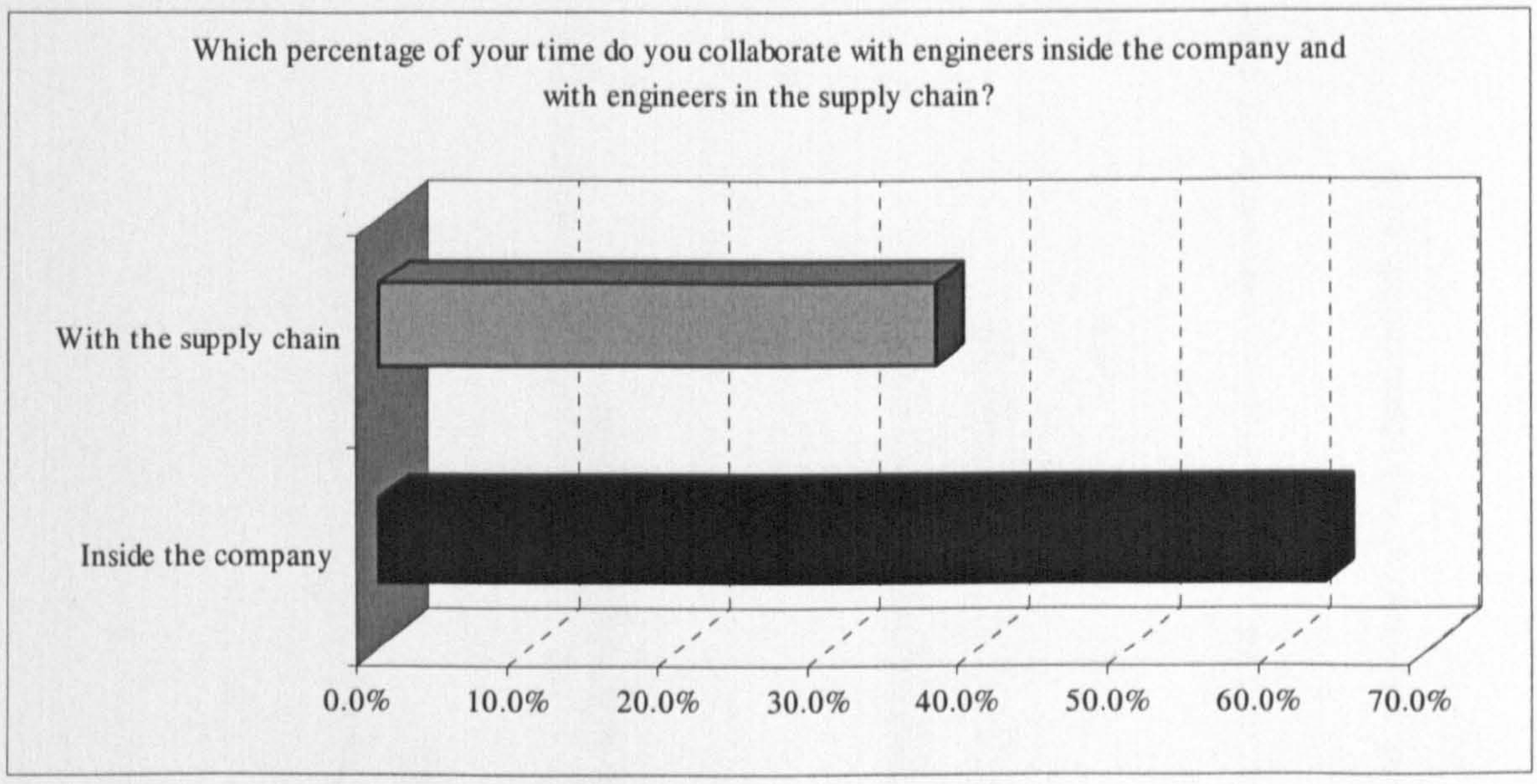


Figure 3.3 Percentage of collaboration time in the extended enterprise

3.3.3 Results from group 3: the required information and knowledge that need to be shared with the supply chain

As explained in section 2.3.6, the knowledge related to product development can be classified in different types: product data, product history and manufacturing constraints. As shown in figure 3.5, manufacturing constraints were deemed important in the following order: design for manufacturing constraints, machine capabilities, and mould design and fabrication constraints. Furthermore, a 20% of the engineers responded that project information is required. These results show that manufacturing constraints are generally preferred as the knowledge to support CPD, rather than product data or product history.

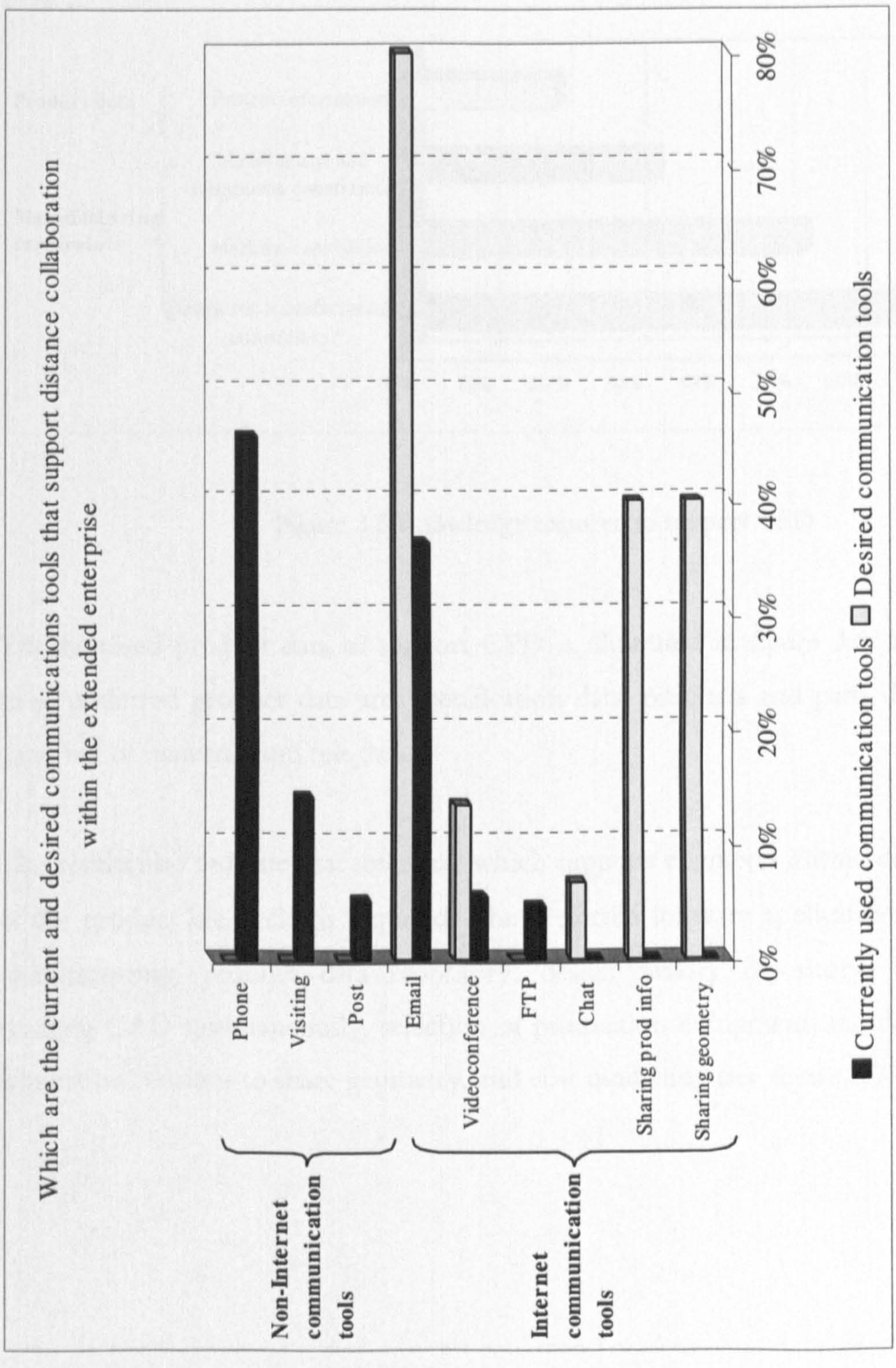


Figure 3.4 Current and desired communications tools

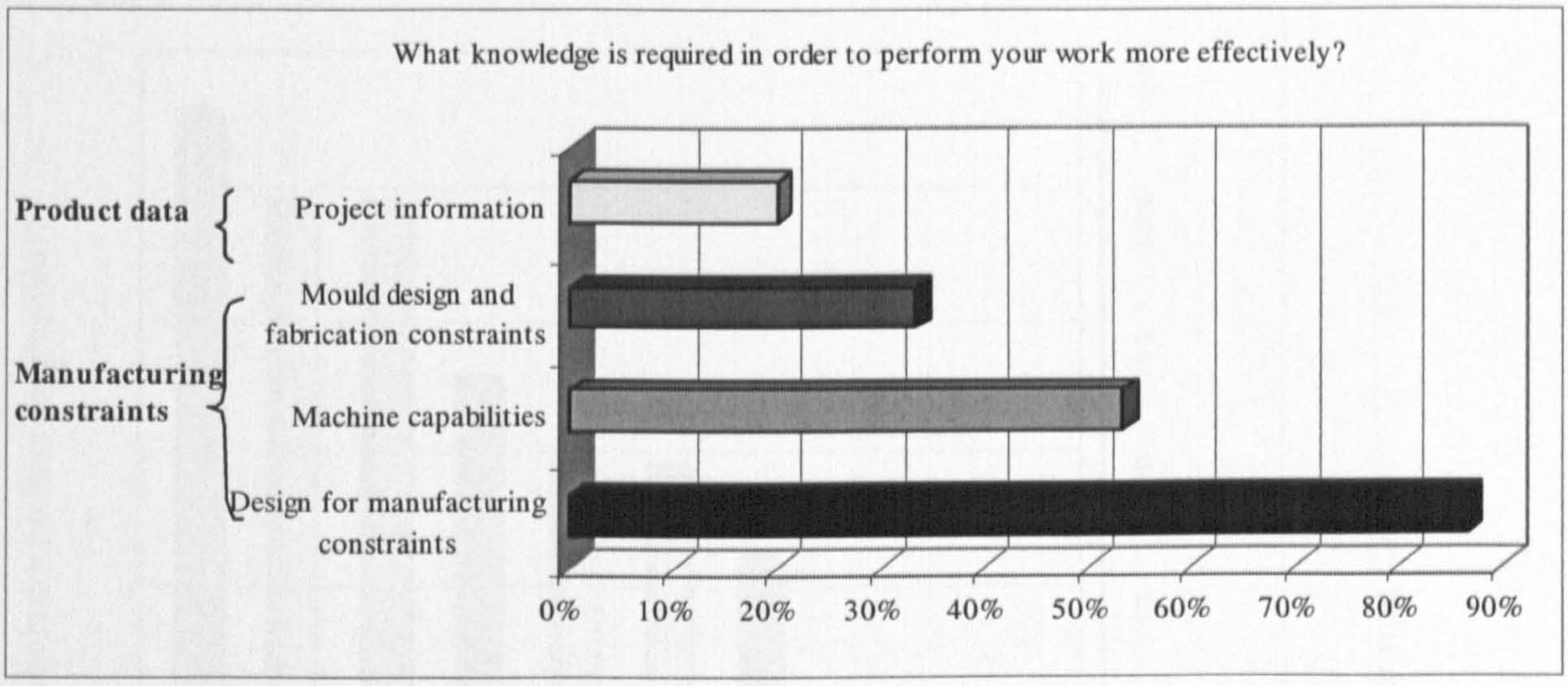


Figure 3.5 Knowledge required to support CPD

The required product data to support CPD is illustrated in figure 3.6. Examples of the most preferred product data are: specification data, products and parts data, geometrical data, bill of materials and test data.

The results also indicate that software, which supports engineers during different activities of the product life cycle, is required. The preferred software applications are: design for manufacturing, product data repository, design history repository, testing services, running CAD simultaneously, selection of production equipment, mould design, mould fabrication, sessions to share geometry, and cost modelling (see figure 3.7).

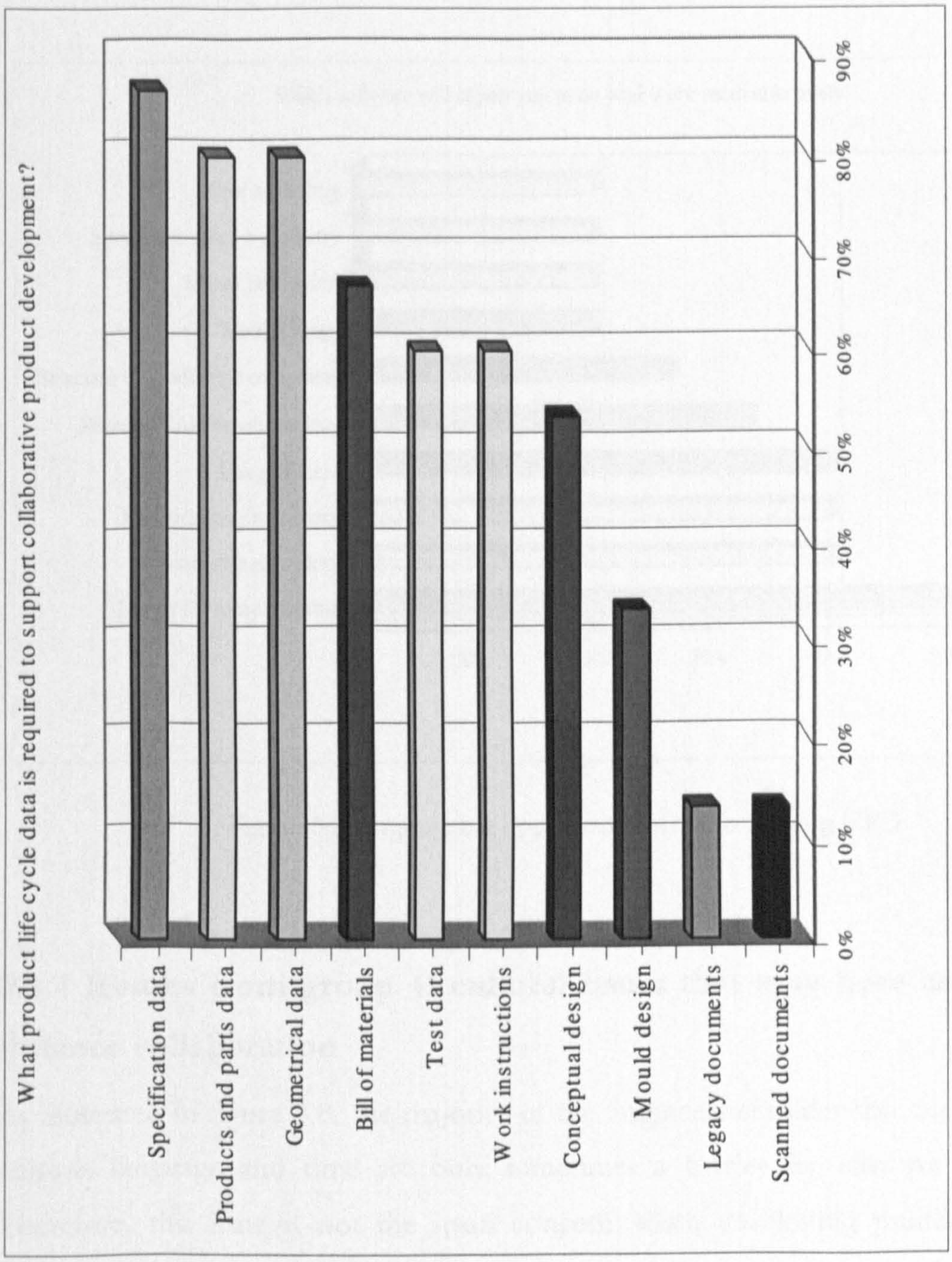


Figure 3.6 Product data required during CPD

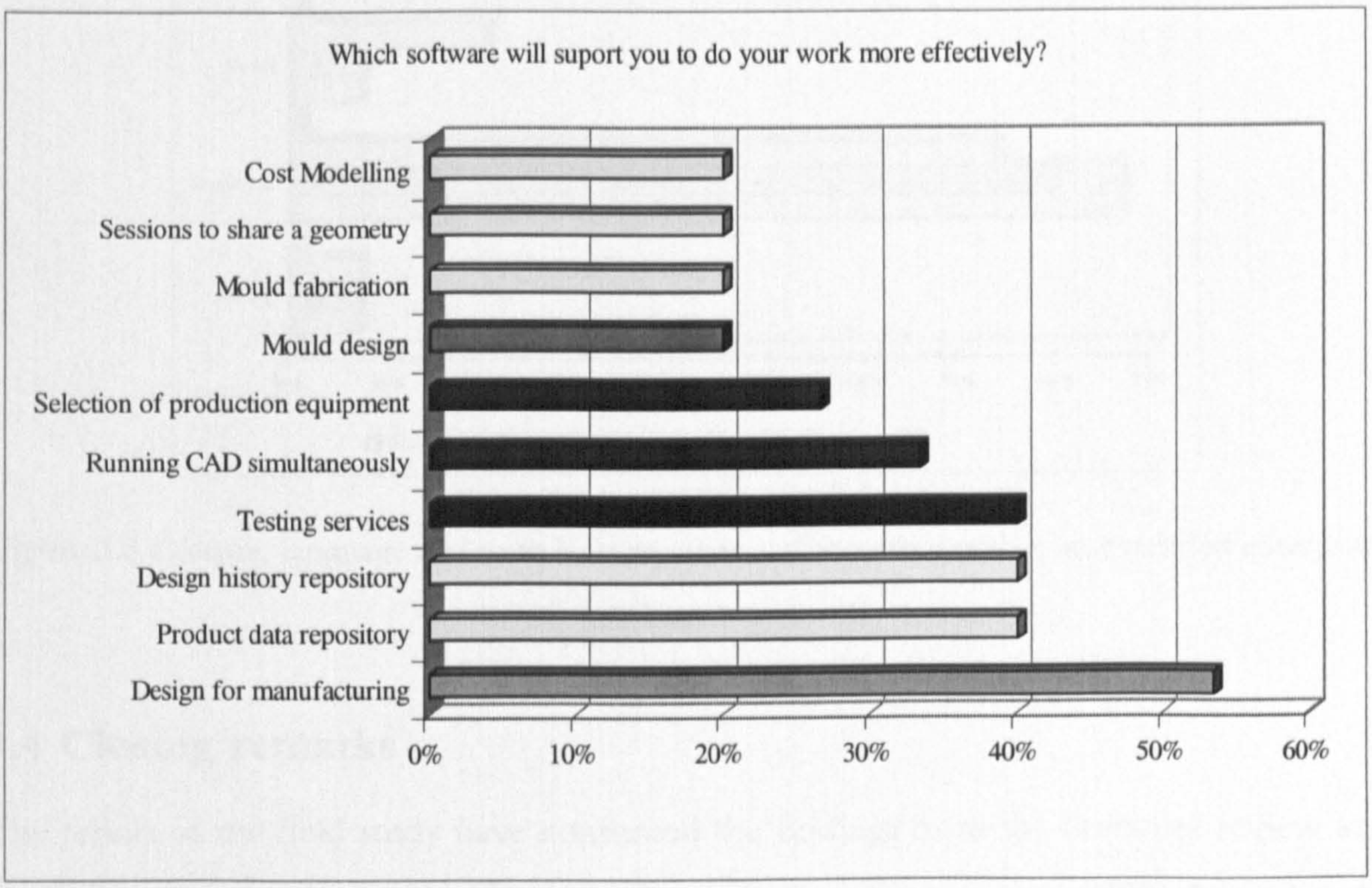


Figure 3.7 Engineering applications preferred during CPD

3.3.4 Results from group 4: cultural issues that may have an impact on distance collaboration

As illustrated in figure 3.8, the majority of the engineers consider that the differences in culture, language and time are only sometimes a barrier for effective collaboration. Therefore, this issue is not the main concern when developing products between a geographically distributed extended enterprise.

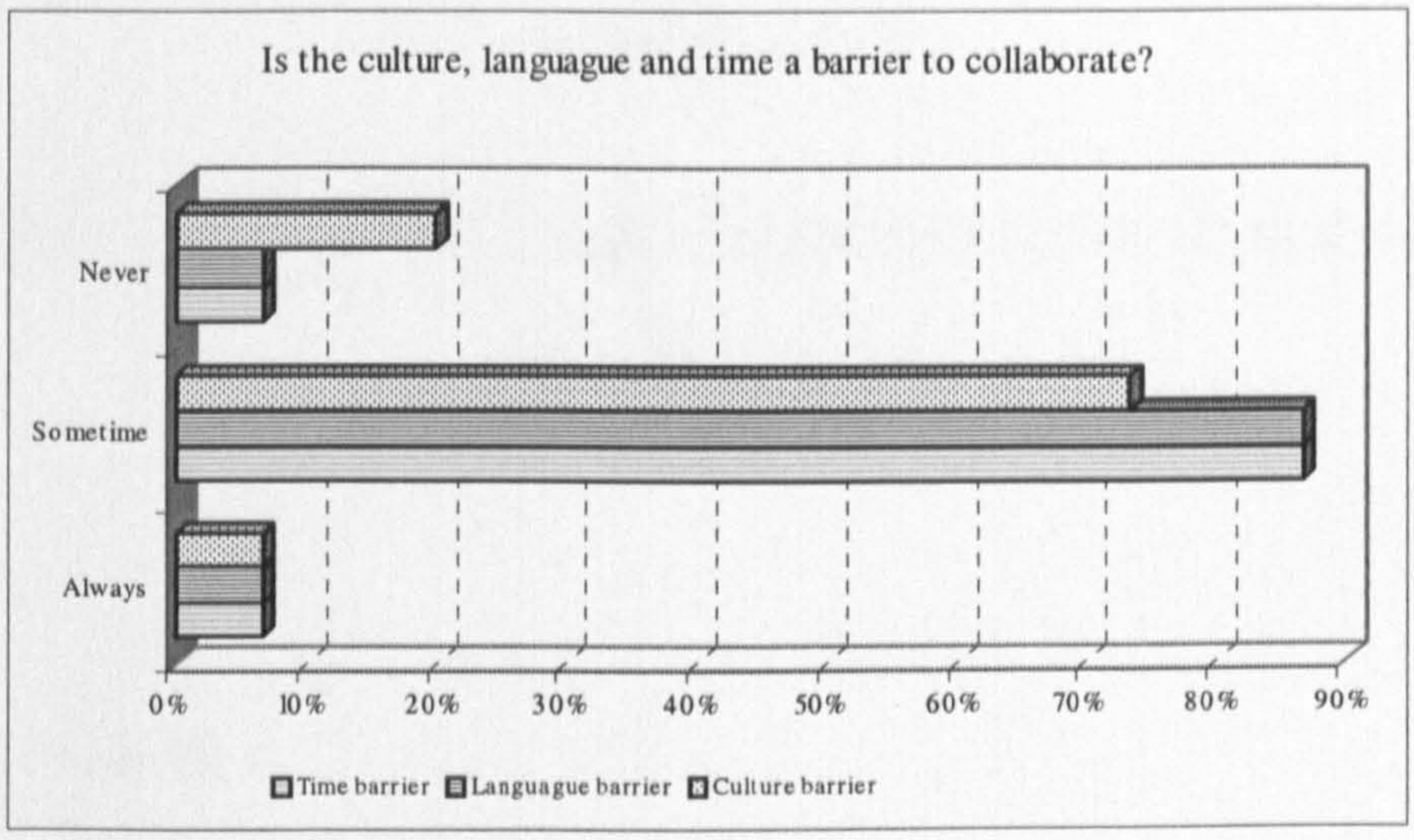


Figure 3.8 Culture, language and time barriers when collaborating within an extended enterprise

3.4 Closing remarks

The results of the field study have confirmed the findings from the literature review and the belief of the author with respect to the criticality of supporting CPD. According to the industry, there is a real industrial need for the development of a CPD system that shares knowledge and information. Furthermore, the study identified a set of requirements that have not been addressed by previously proposed CPD systems. These requirements are:

- Sharing knowledge related to different activities of product development, such as design for manufacturing constraints, machine capabilities and mould design and fabrication constraints.
- Sharing product life cycle data, such as specifications, products and parts data, as well as mould design data.
- Providing engineering applications to support a range of activities of CPD.

Chapter 4

Knowledge Driven Collaborative Product Development

4.1 Introduction

This chapter presents key research issues identified during the literature review and the field study and identifies the author's main contribution in the CPD systems research area. Finally, it describes in detail the reference framework followed to develop a knowledge driven collaborative product development (KdCPD) system architecture.

4.2 Collaborative Product Development research issues

The findings of the field study, described in the previous chapter, were mapped with those ones of the literature survey. This mapping highlighted the criticality of having effective collaboration among the engineers of geographically distributed companies in an extended enterprise. It also reassured the belief of the author that a knowledge driven KdCPD system provides a solution to this need. Such a system requires a distributed, interoperable and secure architecture for its development. It should also fulfil other requirements, which were highlighted from the mapping. These requirements are classified as follows:

1 Addressed research issues: the research community has successfully addressed these issues.

2 Evolving research issues: these issues have been identified but no satisfactory solutions have been given to them yet.

3 Emerging research issues: these issues have not been raised nor addressed in the CPD systems research area.

The following subsections present in detail the above issues.

4.2.1 Addressed CPD research issues

The technological requirement of including communication tools in CPD systems has been found to be the only requirement to be well addressed, as detailed in section 2.3.2. Currently, different communications tools have been well integrated with some of the proposed research and commercial CPD systems in order to provide a better communication among the distributed team.

4.2.2 Evolving CPD research issues

The research community has raised other requirements, which importance has been confirmed by the field study, but for which no completely satisfactory solutions have been achieved as yet. These requirements are the following:

1 Engineering applications to support key collaborative activities

The current CPD systems' software applications are mainly focused on supporting activities that are performed during the product design activity, such as conceptual design or detail design. However, as concluded during the field study, the industry requires a CPD system that supports a wider range of key activities of product life cycle. These applications could become an effective tool, if they provide decision making support. Therefore, they are referred as decision support engineering applications.

2 Product Model

In order to support key activities of the product life cycle, there is a need to capture product data. This requirement could be addressed by having a Product Model, which captures different product life cycle data (i.e. product engineering data, manufacturing and tooling data). Although some research has been conducted in this area, such as the implementation of Product Data Management (PDM) systems, these systems only provide a static view of documents and are mainly concerned with the product

engineering aspect. They, therefore, do not support decision making during different activities of the product life cycle.

3 Geometric representation

The visualisation of virtual geometric models in CPD systems has been addressed by the research community and there are several commercial tools available for visualising geometric models in a collaborative environment. However, two main points should be emphasised in future generations of CPD systems. First, the geographically distributed team should share geometry in such a way that it could be modified in real time; and second, geometric models should be integrated with decision support engineering applications.

4 Project and team management applications

The industrial requirement to coordinate the virtual team and their project's tasks has been addressed through applications to administrate the team members, their information rights and the project workflow. However, these applications are not fully integrated with other decision support engineering applications. Therefore, project and team management in a collaborative environment should be further investigated.

4.2.3 Emerging research issues

1 Capture, representation and provision of product life cycle knowledge

A major requirement that has not been considered by the research community is the capture, representation, and provision of product life cycle knowledge in the time, place and format required to support engineering decision making in a collaborative environment.

Based on the literature review of section 2.3.6 and the field study, the author believes that the suitable type of product life cycle knowledge is the one referred to as manufacturing constraints. This type of knowledge refers to the various considerations that limit the decisions that can be taken during product development. It is considered to

be the most suitable because it does not only capture information of previous and current product development projects, but it actually constrains the decisions that can be taken based on certain limitations, such as process and resources capabilities. Therefore, in this research the words product life cycle knowledge and manufacturing constraints are used interchangeably to refer to the type of knowledge identified in section 2.3.6 called manufacturing constraints.

The product life cycle knowledge also requires to be represented in such a way that the impact of one manufacturing constraint on other engineering activities is highlighted. This cannot be achieved without capturing the relationship between the manufacturing constraints of the different activities of the product life cycle. This unique characteristic of the knowledge representation is subsequently referred to as knowledge integrity.

The geographic distribution of the knowledge among the companies of the extended enterprise is another issue that requires to be considered. The manufacturing constraints that affect the decisions that are taken during the activities performed by the different companies of the extended enterprise should be retained inside each company while still maintaining its knowledge integrity. Thus, the companies' intellectual property is captured and documented in a knowledge database.

Based on the previously explained research issues, this research is proposing a knowledge driven system architecture to support CPD. The following sections describe in detail the reference framework that was used to develop such a KdCPD system.

4.3 Reference framework to develop a Knowledge driven Collaborative Product Development (KdCPD) system

In order to develop a KdCPD system architecture addressing the research issues described in the previous section, it was necessary to have a reference framework to guide its development. The reference framework aims to provide a clear understanding of the different views associated with an extended enterprise in order to be used as a basis for

the system development. The representation of these views, such as what activities are performed in an enterprise, how, when, where and by whom is also known as an enterprise model (Zachman 1997). These views are represented through models by using different modelling techniques, such as information modelling or process modelling. A model is a simplified representation of a system at some particular point in time (Bellinger 2003).

In this research, CIMOSA (ESPRIT Consortium AMICE 1993) was used as a reference framework because it was considered to be a clear, flexible and widely used framework to model different views of an extended enterprise. The primary objective of CIMOSA is to provide a reference for analysing the evolving requirements of an enterprise and for translating them into a system that enables and integrates the functions to meet the requirements.

CIMOSA reference framework, illustrated in figure 4.1, supports the representation of enterprise requirements, the system architecture design, as well as the implementation of the system (see figure 4.1-a). These representations are done through the definition of four different views, which could be modelled using different enabling techniques (see figure 4.1-b). The modelling of different views complies with the common practice of focusing on different aspects of an enterprise, rather than looking at an enterprise as a whole. The views defined by CIMOSA (organisation, resource, information and process) were adapted in this research to completely describe the requirements of an extended enterprise and not only of one company.

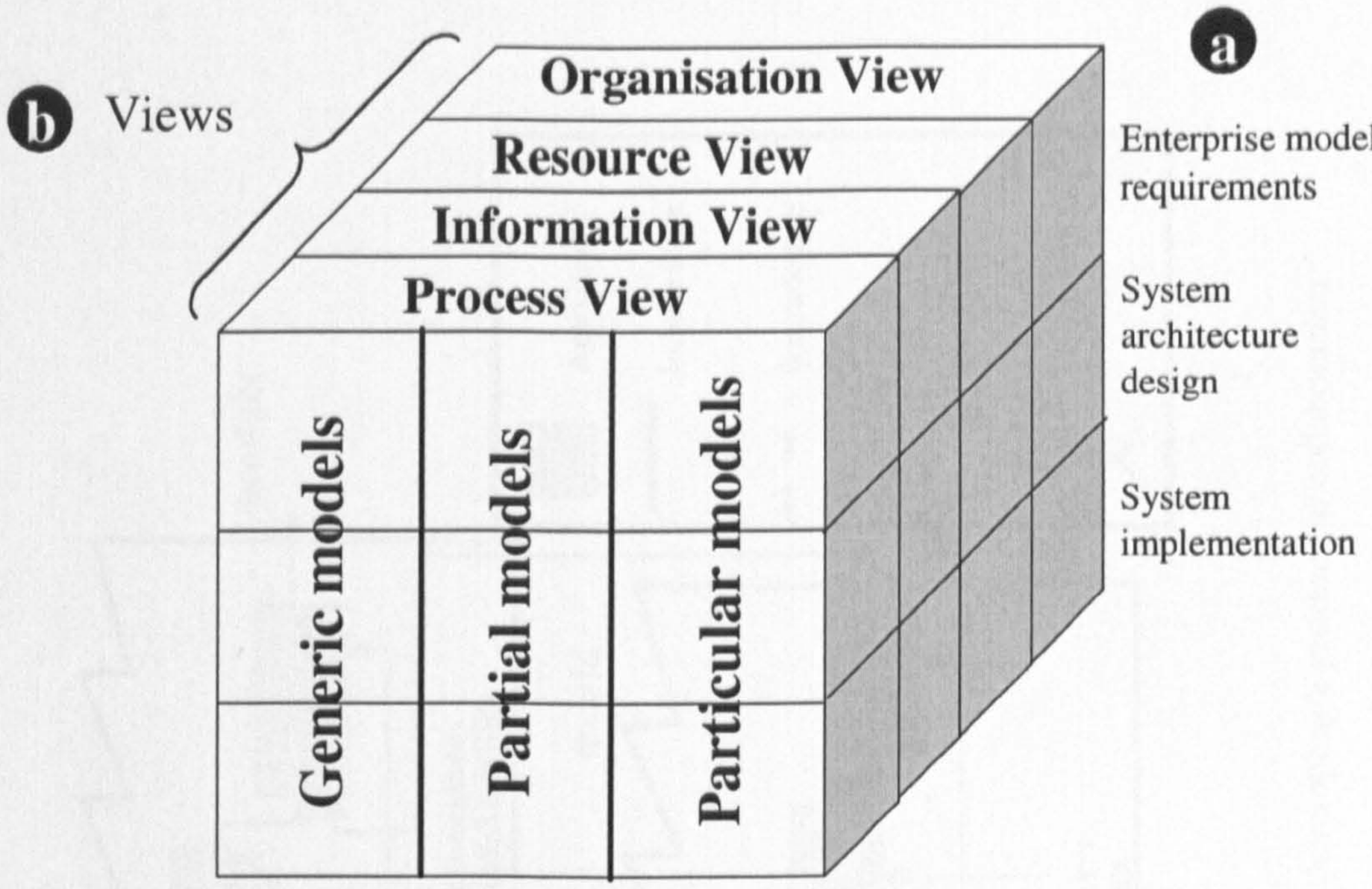


Figure 4.1 CIMOSA reference framework (ESPRIT Consortium AMICE 1993)

Figure 4.2 illustrates the different views of an extended enterprise that were modelled. As the figure shows, activities (activity view) are developed inside different companies, which are geographically distributed (location view). The engineers (organisation view) work together in a virtual team to perform these activities. For this, they exchange product information (information view) using different communication mechanisms. Moreover, various limitations (knowledge view) should be taken into account when taking decisions to perform these activities.

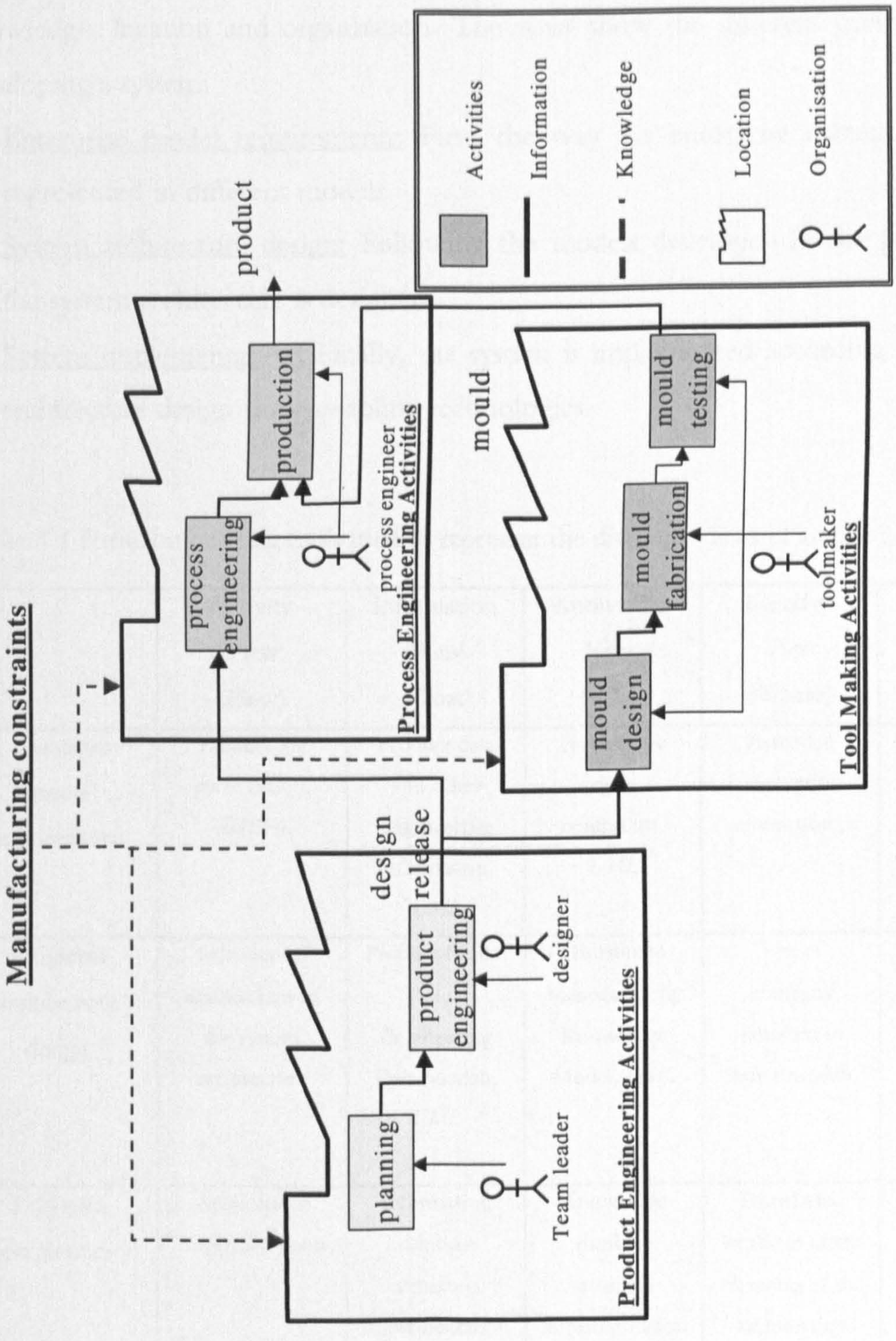


Figure 4.2 Different views of an extended enterprise when developing a product in collaboration

The columns in table 4.1 show the views of the extended enterprise that were modelled according to the CIMOSA reference framework. These are: activity, information, knowledge, location and organisation. The rows show the different perspectives when developing a system:

1. Enterprise model requirements: First, the way the enterprise currently operates is represented in different models.
2. System architecture design: Following the models developed in the previous stage, the system architecture is designed.
3. System implementation: Finally, the system is implemented according to the system architecture design using enabling technologies.

Table 4.1 Formal modelling tools used to represent the different views of an extended enterprise

	Activity View (How)	Information View (What)	Knowledge View (Why)	Location View (Where)	Organisation View (Who)
1. Enterprise model requirements	Product life cycle (PLC), *IDEF0*	Product data and other engineering information, *UML*	Product life cycle constraints, *UML*	Extended enterprise interactions,	Engineers roles, skills and security issues
2. System architecture design	Selection of applications of the system architecture	Product Model and Engineering Data Models, *UML*	Distributed Manufacturing Knowledge Model, *UML*	Use of company notation in other models	Organisation Model to manage team and security issues, *UML*
3. System implementation	Applications implementation	Information database structure implementation *classes, attributes and methods*	Knowledge database structure implementation *classes, attributes and methods*	Distributed locations of the elements of the architecture	Organisation database structure implementation *classes, attributes and methods*

CIMOSA reference framework gives the flexibility to use any modelling technique adequate to the requirements of the system being developed. For this research, the modelling tools that were used to model each view of the extended enterprise are also shown in table 4.1.

The following subsections describe in detail the different views and the selected modelling techniques.

4.3.1 Activity view

In order to understand the information and knowledge requirements of the extended enterprise during CPD, it was firstly necessary to have a clear view of how product development activities are currently performed. This model was useful for the identification of information and knowledge driven manufacturing activities.

A number of techniques are available to model activities or processes of an enterprise. Examples are: IDEF0 (Colquhoun, Baines et al. 1993), IDEF3 (Mayer, Menzel et al. 1995) and the diagram flow. In this research, IDEF0[1] was used to formally represent the key activities of CPD. IDEF0 stands for ICAM Definition Level 0 (Colquhoun, Baines et al. 1993), and is a top-down hierarchical method that describes a system by a series of functions arranged sequentially as shown in figure 4.3. The hierarchical breakdown allows the definition of a system in different levels of detail. Moreover, this enabling technique was good and clear in the representation of:

1 Key activities of geographically distributed product development.
2 Information input and output for each activity.
3 Mechanisms required to perform each activity.
4 An abstract view of the knowledge that control or constrain each of the activities.

[1] See appendix B for an overview of IDEF0 technique

A drawback of this technique is that the IDEF0 notation cannot represent the location where the activities are performed. To overcome this, the author introduced a special notation to represent a company location. This notation is illustrated in figure 4.3 and was used to group activities performed inside a particular company. The activity modelling is presented in chapter 5.

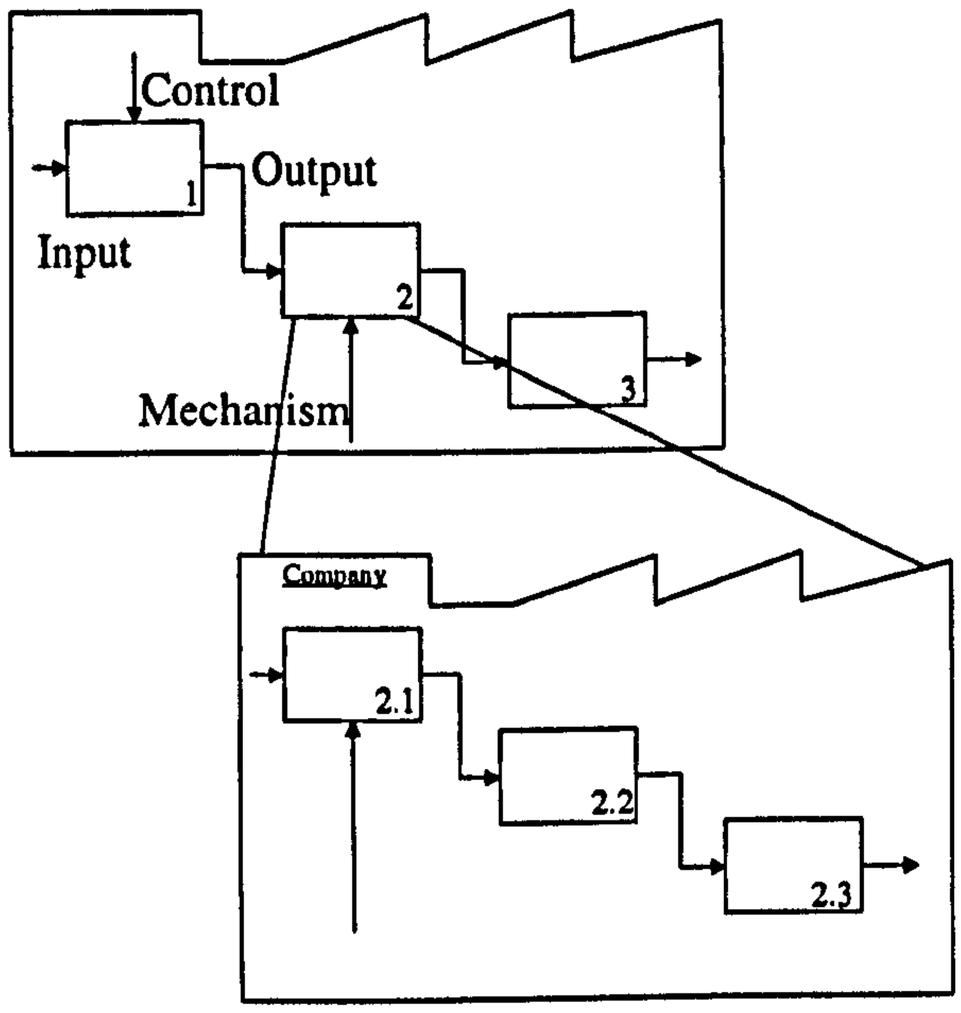


Figure 4.3 The hierarchy of an IDEF0 model

4.3.2 Location view

This view is not part of CIMOSA reference framework but it was included because it was necessary to represent the geographically distributed nature of product development. As explained in section 4.3.1, a special notation was used to represent companies in geographically distributed locations. This approach is used throughout the development of the KdCPD system architecture.

4.3.3 Knowledge view

The author agrees with others (Al-Ashaab 1994; Molina 1995; Young, Wang et al. 2001) that one of the fundamental issues that must be addressed in order to provide decision making support during product development, is the identification of an appropriate

representation of the knowledge. This representation has been called manufacturing model, which has been defined as "a model that captures the information that describes the characteristics and the resources of the process as well as the constraints which govern the use of the process" (Al-Ashaab 1994). The research efforts in this area (Al-Ashaab 1994; Molina 1995; Young, Wang et al. 2001) have shown promising results as a basis for the capture and representation of structured knowledge to support decision making. However, the use of this model as a basis to support decision making during CPD amongst a geographically distributed extended enterprise, which has been found critical, has not yet been addressed. Therefore, this research extends the manufacturing model concept in order to address this issue. Furthermore, it captures the knowledge in the form of "Manufacturing Constraints" as explained in section 4.2.3. The manufacturing model will be referred to as Manufacturing Knowledge Model in this research and it is presented in chapter 7.

Furthermore, the research addresses the following manufacturing modelling research issues:

1 What knowledge must be included in the Manufacturing Knowledge Model to support decision making during CPD?
2 Where is the knowledge located?
3 How could manufacturing constraints be captured?
4 How could the geographically distributed manufacturing constraints be represented in the Manufacturing Knowledge Model?
5 How could the Manufacturing Knowledge Model provide a common source of integrated knowledge to support CPD?

In order to build such Manufacturing Knowledge Model, the following techniques are available:

1. Knowledge representation languages
2. Object oriented languages

These techniques will be further discussed in the following paragraphs.

Knowledge representation languages

These languages are concerned with representing knowledge as facts and rules about a particular subject. An example is an ontology, which is a specification of the objects and concepts that are assumed to exist in a domain of interest and the relationships and semantics constraints between them (Knublauch and Rose 2000). A disadvantage of this technique is that the different components of the system architecture are typically implemented using languages different from the knowledge representation language (Gomez-Perez and Benjamins 1999; Knublauch and Rose 2000). Consequently, additional software, such as translators between knowledge representation languages and system implementation programming languages, are required.

Object oriented languages

The knowledge can be viewed as information plus processing, just as information can be viewed as data plus meaning. Thus, a modelling technique that can represent a mechanism to process information could be used to model knowledge. In order to clarify this concept, figure 4.4 illustrates the difference between data, information and knowledge. As illustrated in figure 4.4-a the number 4 is mere data, and has no meaning in itself. When it is established that 4 mm is the diameter of a cooling system passage of a mould then this value converts to be information (see figure 4.4-b). Furthermore, there is knowledge that establishes that the diameter of a cooling passage must be in the range of 6 to 10mm. Processing this knowledge it can be established that the diameter of 4 mm is not suitable to be used in a mould (see figure 4.4-c).

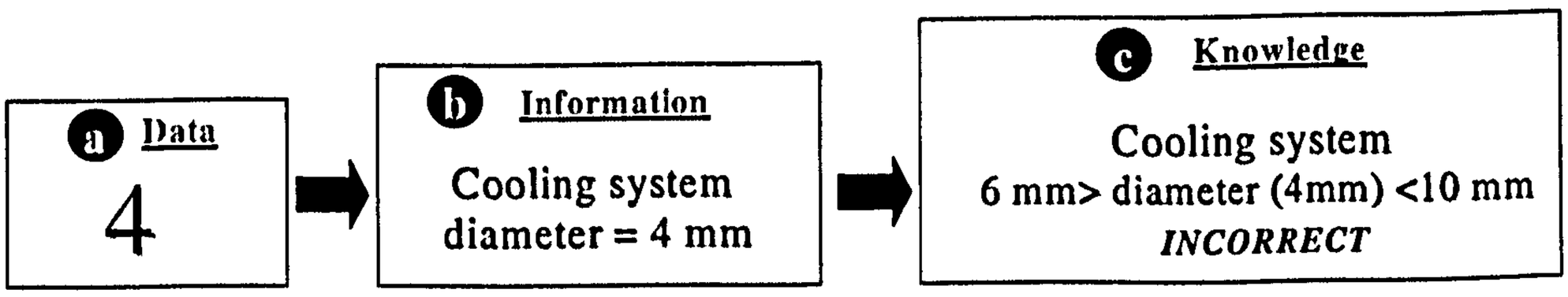


Figure 4.4 Difference between data, information and knowledge

In an object oriented language, an object is defined by its attributes and methods. Attributes describe the characteristics of an object and the methods describe its behaviour. Methods can be used to capture manufacturing constraints (knowledge), as they are able to capture a constraint in the form of a rule, or a mathematical formula. The manufacturing constraints can then be applied on information, in real time, producing a result. For example, figure 4.5 shows how the constraint for the cooling system diameter is captured in a method of a class called "Cooling System Diameter Constraint". When this rule is applied on the information of the cooling system, a result is produced. In this case, the method produces as a result a message stating that the information is incorrect, as the diameter of the cooling system is not suitable to be used in a mould.

Object oriented languages are also strongly related to formal knowledge structures like ontologies as they represent an abstract view of the world. The advantage of this technique is that it can eventually be transformed into an implementation language, which can be linked to other information and applications.

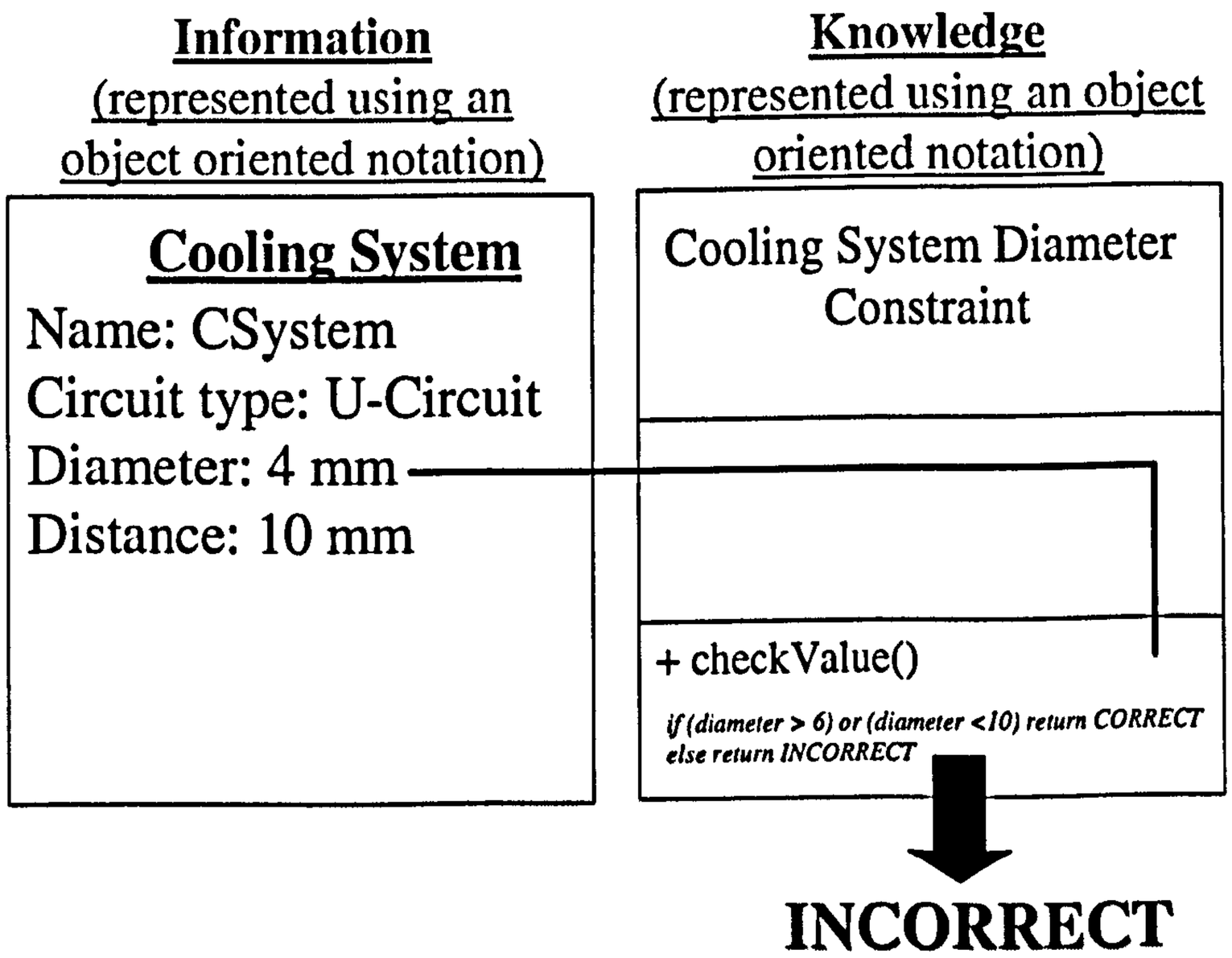


Figure 4.5 Knowledge representation using an object oriented notation

In this research, the Unified Modelling Language (Rumbaugh, Jacobson et al. 1999) was used to formally represent the manufacturing constraints. UML[2] is a framework for describing a set of models to represent any complex system (Sommerville 2001). In particular, the class model was used to represent an unambiguous representation of the manufacturing constraints organised into a taxonomy. This language was selected for the following reasons:

1 UML notation is clear and easy to understand and it is the industry standard formal modelling language for developing object oriented databases.
2 UML has an object oriented approach, which was required to represent the complex nature of the knowledge and the interactions among manufacturing constraints.
3 UML technique can be used to produce information and knowledge models, which can be reused during the object oriented design of the KdCPD system architecture.
4 The representation is flexible enough for future customisations and upgrades.
5 The notation provided by the language is useful to structure the knowledge according to the knowledge classification identified during the activity modelling. As such, the class notation, shown in figure 4.6-a, was used to represent manufacturing constraints and the "package" symbol, shown in figure 4.6-b, was used to group these manufacturing constraint classes according to the product development key activities that require knowledge support. A package is a modular component containing two or more classes or subpackages.

The representation of the geographical distributed nature of the knowledge was addressed by using the location notation. This notation was introduced during the activity modelling and it is used to group packages which represent the knowledge required by a company.

[2] See appendix D for an overview of UML

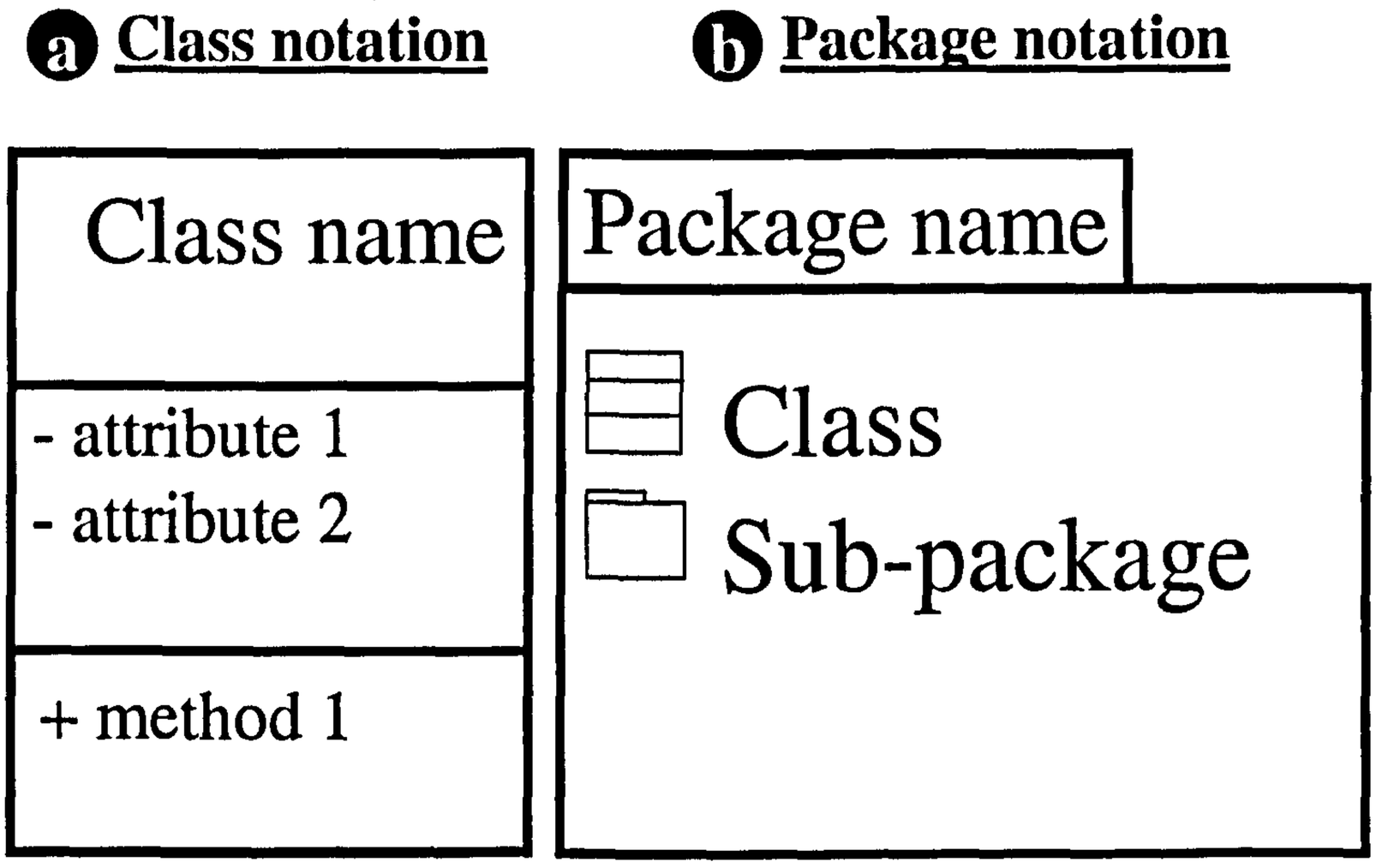


Figure 4.6 UML notation of a class and a package

4.3.4 Information view

The information required to support CPD was represented in a Product Model, as discussed in section 4.2.2. Other information required to support decision making, such as plastic information, company resources and mould standard parts information, were also represented in information models. A detailed explanation of these models is given in chapter 7.

In this research, the UML object oriented language (Rumbaugh, Jacobson et al. 1999) was used to represent the Product Model. This language was selected because the information models can be directly mapped to the design and implementation of the databases in the system architecture. It also provides an unambiguous representation of the information using classes and relationships organised in a taxonomy. As such, product information, such as part features, mould description and selected resources, were represented and interrelated based on two main relationships:

1 **Aggregation:** These relationships are used to represent how a class is composed of other classes (see figure 4.7). For example, a part is composed of a set of features.
2 **Generalisation:** These relationships are used to represent a hierarchical organisation in the taxonomy, where the most general classes are presented in the top of the hierarchy. More specialised classes inherit their attributes and methods. These classes may also have their own attributes and methods. For example, the "Feature" class, shown in figure 4.7, has a set of attributes that are common to all features, such as position and criticality. Furthermore, the inherited "Wall" class, shown in figure 4.7, has attributes such as length plus the attributes inherited from the "Feature" class.

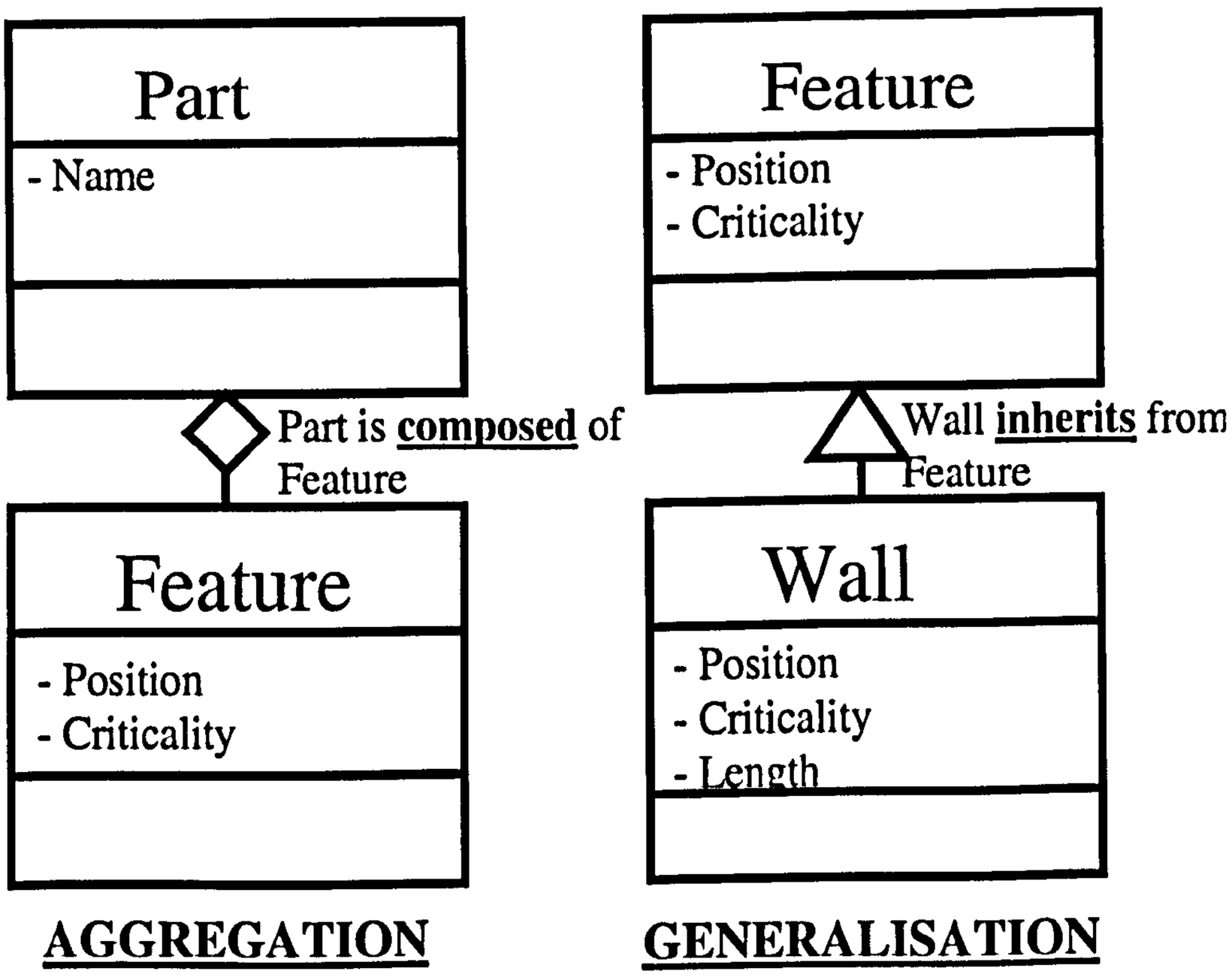


Figure 4.7 UML notation of the aggregation and generalisation relationships

4.3.5 Organisation view

The organisation view is concerned with the way in which the geographical distributed team members are organised, their information requirements and other security issues.

For this purpose, UML object oriented language (Rumbaugh, Jacobson et al. 1999) was used. The Organization Model is presented in detail in chapter 7.

4.4 Closing remarks

In conclusion, while previous research has focused on some requirements of CPD systems, this research is proposing a KdCPD system which main innovation is the capture, representation and provision of knowledge to support decision making throughout the different activities of product development.

In order to build such KdCPD system architecture, which addresses the main innovation, it was critical to have an overview of the extended enterprise. CIMOSA proved to be a good guide to model the different views of the extended enterprise. Such views are: activities, location, knowledge, information and organisation. The modelling of these views provided a basis for designing and implementing the system architecture.

Chapter 5

Activity Modelling in a Collaborative Environment

5.1 Introduction

The development of a knowledge driven system architecture to support decision making during collaborative development of injection moulded products was founded on the development of a clear overview of the activities and information flows involved. This chapter describes the activity modelling performed to represent the current injection moulding product development amongst an extended enterprise. The activity modelling produced a model, which represents the activities performed by an injection moulding company and its collaborators.

The activity model highlighted the need to support different engineering activities, which need to be performed in collaboration. Furthermore, it provided the starting point for the identification of the required knowledge and information throughout CPD. Based on this, the knowledge and information modelling was performed as presented in chapter 7.

5.2 Methodology to develop the activity modelling

The activity model was developed based on information gathered during interviews with some of the engineers whom were approached during the field study. A description of their activities within their company was given during face to face interviews. The activity model was developed using IDEF0 (Colquhoun, Baines et al. 1993) modelling technique (see section 4.3.1) and its accuracy was later verified with the engineers. Furthermore, problems that occur during product development were also exposed.

Engineers of different departments were interviewed to obtain different perspectives of the activities performed during CPD. These departments are: sales and marketing, purchasing, project management, design, production, quality and inspection.

5.3 Activity modelling of injection moulded Collaborative Product Development (CPD)

Figure 5.1 illustrates a scenario of the development of an injection moulded product between an extended enterprise. Initially, the customer or OEM selects a supplier company to develop and produce an injection moulded product, which is needed as a subcomponent of their final product. The supplier company, which is referred to as "Product Engineer", designs and produces the injection moulded product in collaboration with the customer, a toolmaker, a process engineer, and other second tier suppliers. These companies are located in different cities or even in different countries, causing challenges to the collaboration throughout the project.

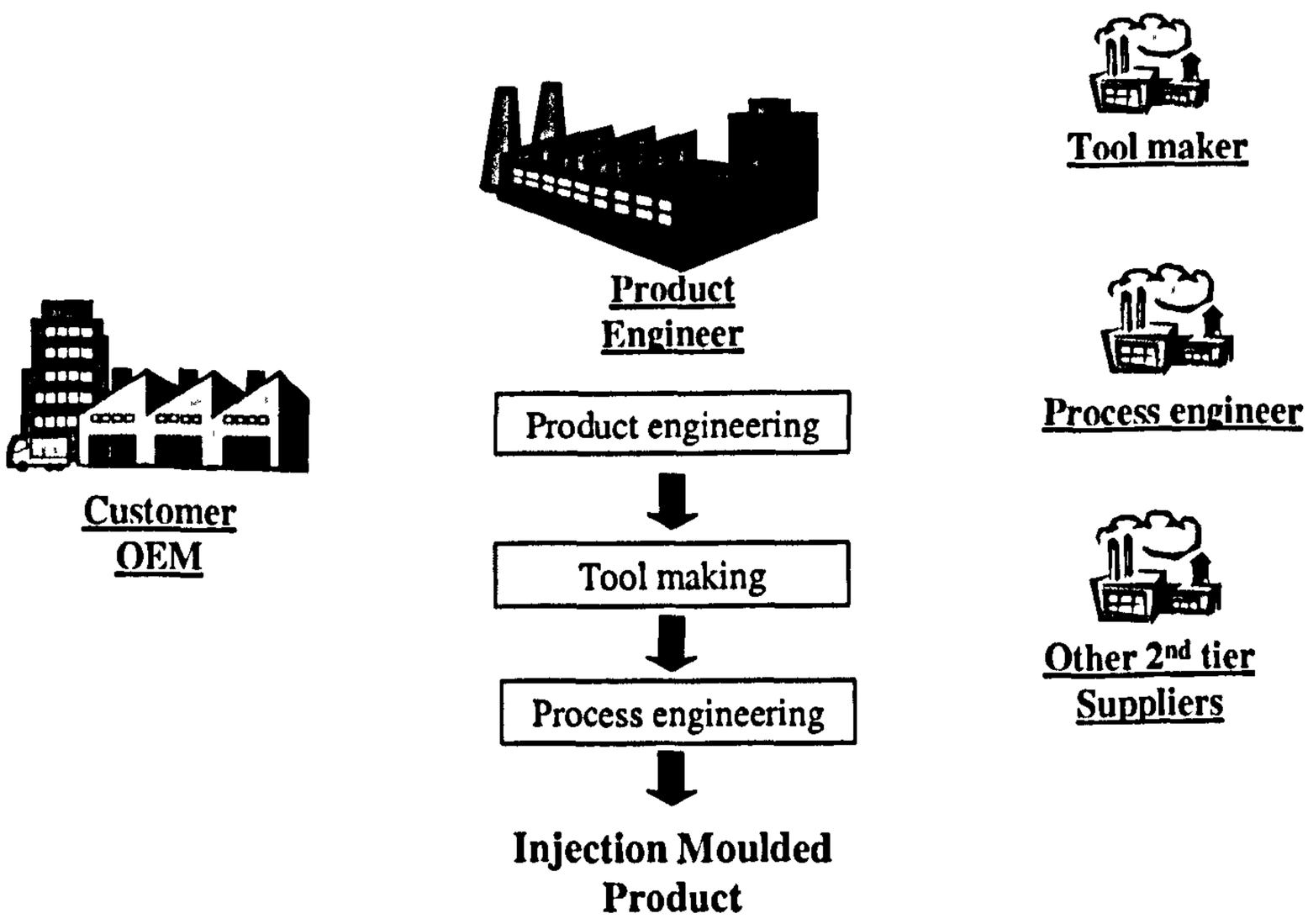


Figure 5.1 Scenario of injection moulded collaborative product development

The top level of the resulting IDEF0 model of collaborative injection moulded product development is shown in figure 5.2. This model illustrates the activities that are performed by the product engineer, toolmaker and process engineer. As discussed in section 4.3.1, the location notation was used to represent the locations were the activities are performed. The complete IDEF0 model is presented in Appendix C.

The model also illustrates the information flows and feedbacks, which are exchanged between the partners, and represent the information and knowledge that affect decision making during the geographically distributed activities. For instance, the output "feedback from toolmaker" affects the decisions taken during design and development. This feedback means there is some iteration before releasing the final product design information.

In addition, the model illustrates manufacturing constraints controlling most of the activities of product development. The companies may or may not posses this knowledge, but it will eventually constrain the right engineering decisions that can be taken. For example, the "feedback from tool making" is caused because there are limitations imposed on the product design, of which engineers become aware during the mould design. These limitations need to be considered at the right time, or they will cause iterations to solve the particular issue.

The next subsections will describe in more detail the main activities of CPD focusing on the activities that have to be performed in collaboration and which information and knowledge is involved.

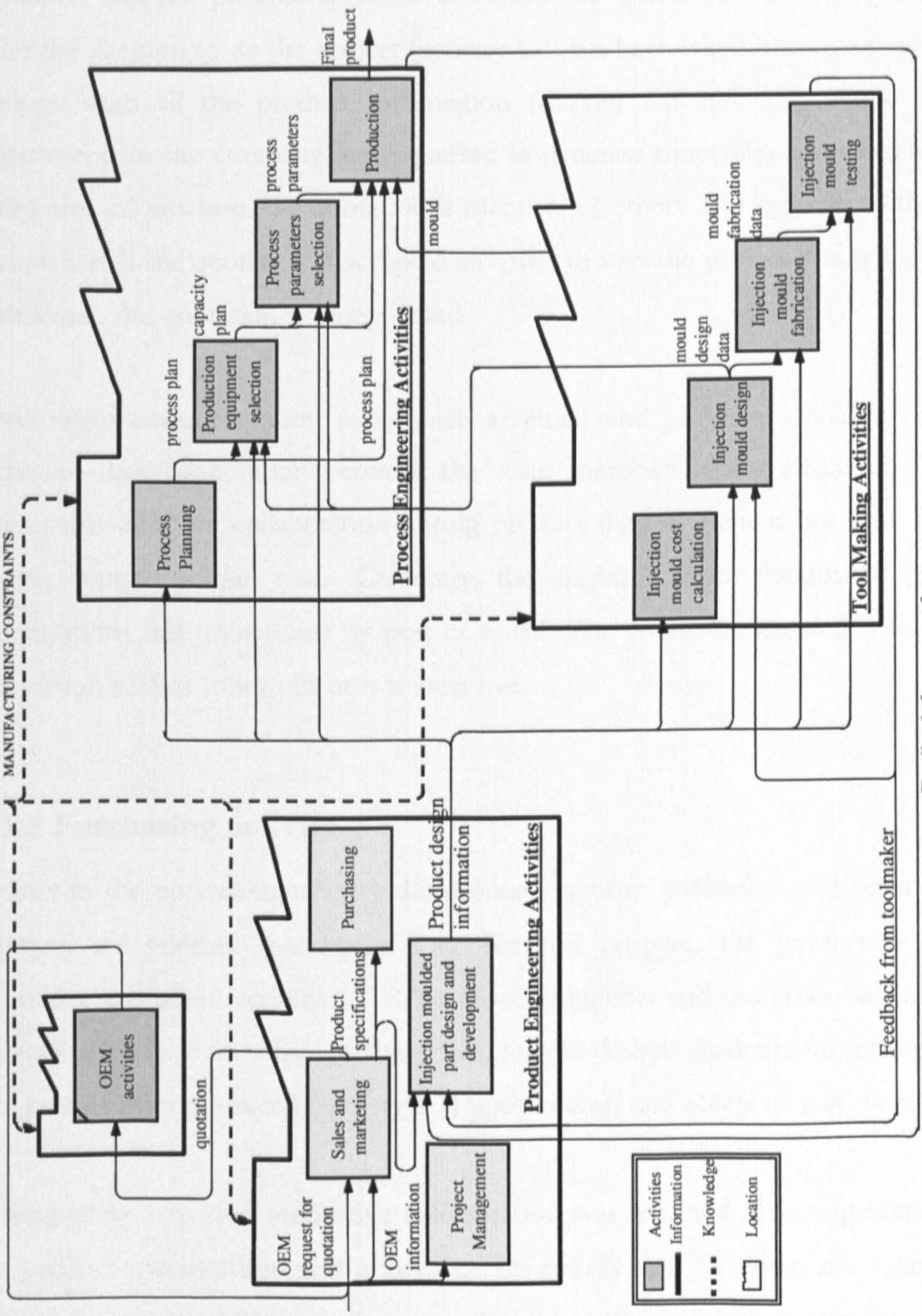


Figure 5.2 Top level activity model which represents injection moulding collaborative product development

5.3.1 Sales and marketing activity

Sales and marketing are the activities where the product engineer receives a request for quotation and the product specifications from the OEM (activity 1.1, see figure 5.3). After the decision to do the project (activity 1.3) has been taken, the company produces a package with all the product information (activity 1.2 and 1.4). Based on this, all departments in the company are requested to generate timetables and allocate resources. Using this information, the quotation is calculated (activity 1.5) and sent to the OEM for acceptance. If the quotation is accepted an order to start the project is raised (activity 1.6). Otherwise, the quotation is renegotiated.

Even when communication tools, such as email and post, are available during these activities, the collaboration between the team members is not effective. This finding proves that effective collaboration during product development is not only achieved by having communication tools. Currently, the engineers share documents (i.e. product specifications and quotations) by post or email. The engineers stated that frequently the submission of data is delayed or it is even lost.

5.3.2 Purchasing activity

As part of the purchasing activity the process engineer, toolmaker and other second tier suppliers are selected (see figure 5.4). For this purpose, the purchasing department requests a quotation (activity 5.1). The process engineer and toolmaker are also provided with additional specifications (activity 5.2), such as design requirements for the mould. A purchase order is produced (activity 5.3) upon receipt and acceptance of the quotation.

During these activities, ineffective collaboration was reported. The engineers stated that the product specifications and quotations are usually sent by email and there is no real time communication between the companies, causing delays and missing data.

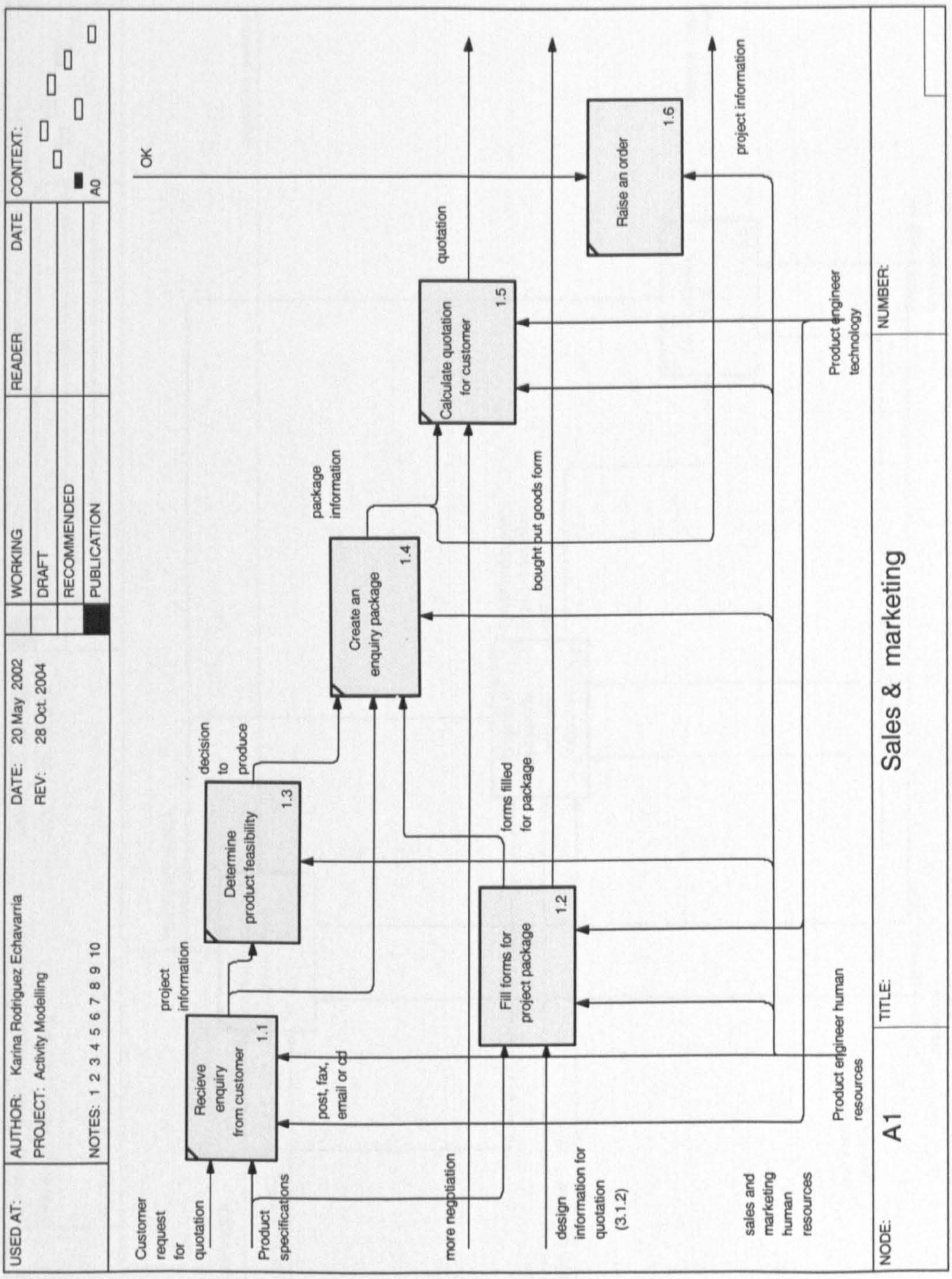


Figure 5.3 Sales and marketing activity

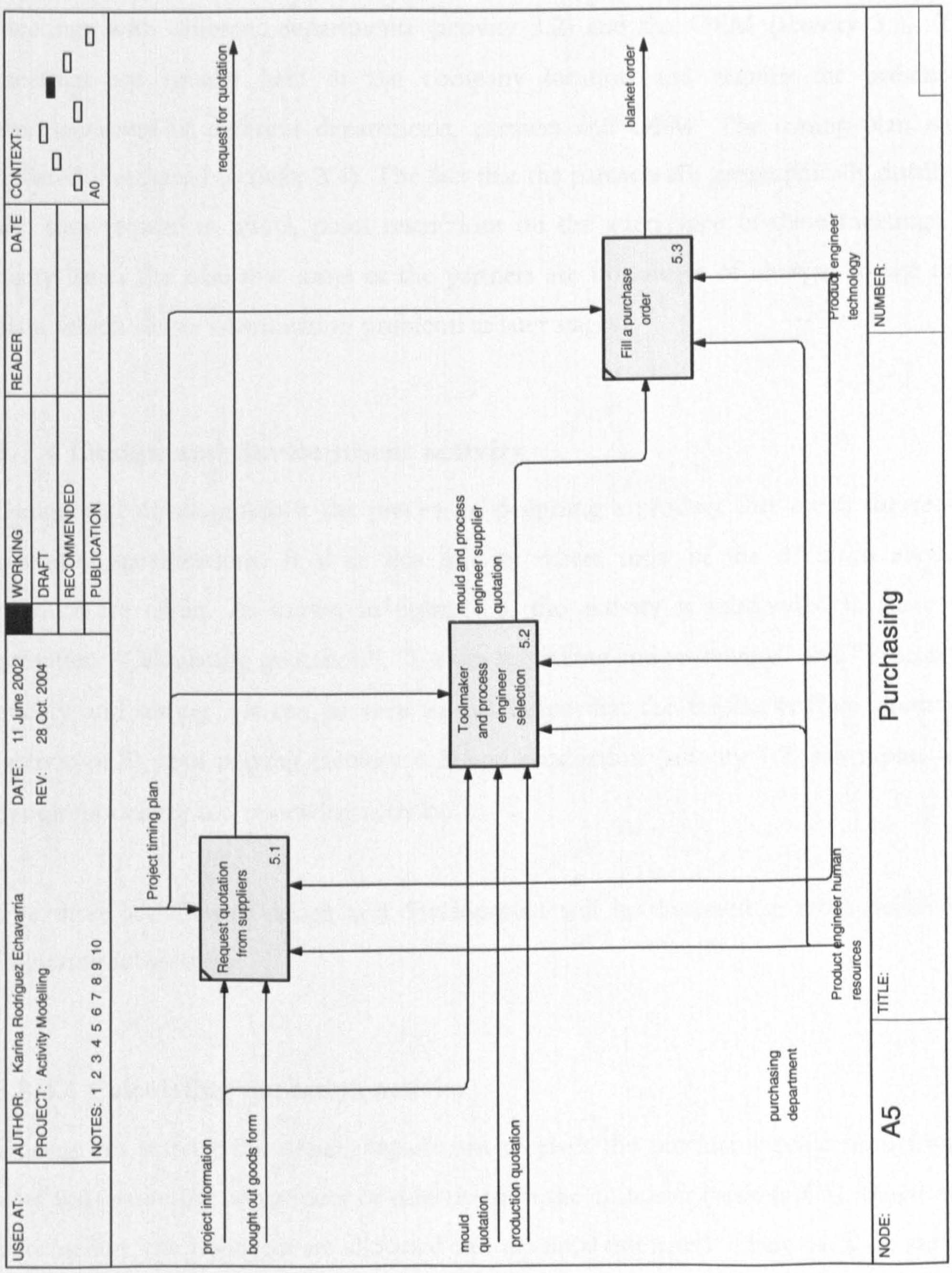


Figure 5.4 Purchasing activity

5.3.3 Project management activity

As shown in figure 5.5, the timing plan is developed (activity 3.1) and reviewed in meetings with different departments (activity 3.2) and the OEM (activity 3.3). These meetings are usually held in the company location and require the presence of representatives of different departments, partners and OEM. The timing plan is then updated if required (activity 3.4). The fact that the partners are geographically distributed, and thus require to travel, poses restrictions on the attendance of these meetings. It is many times the case that some of the partners are not aware of changes to the timing plan, which causes coordination problems at later stages.

5.3.4 Design and development activity

Design and development is the process of designing a product that meets the required customer specifications. It is in this activity where most of the decisions about the product are taken. As shown in figure 5.6, the activity is subdivided in three main activities: "Calculating quotation", "Design modelling and reviewing" and "Prototyping, quality and testing". It can be seen in this figure that the feedbacks from prototyping (activity 4.3), tool making (activity 6.2) and production (activity 7.7) are inputs to the design modelling and reviewing activity.

The three activities of design and development will be discussed in more detail in the following subsections:

5.3.4.1 Calculating quotation activity

During this activity the design department receives the product specifications from the sales and marketing department or directly from the customer (activity 4.1). Based on this information, the resources are allocated and the costs estimated. These costs are then sent to the sales and marketing department for producing the quotation.

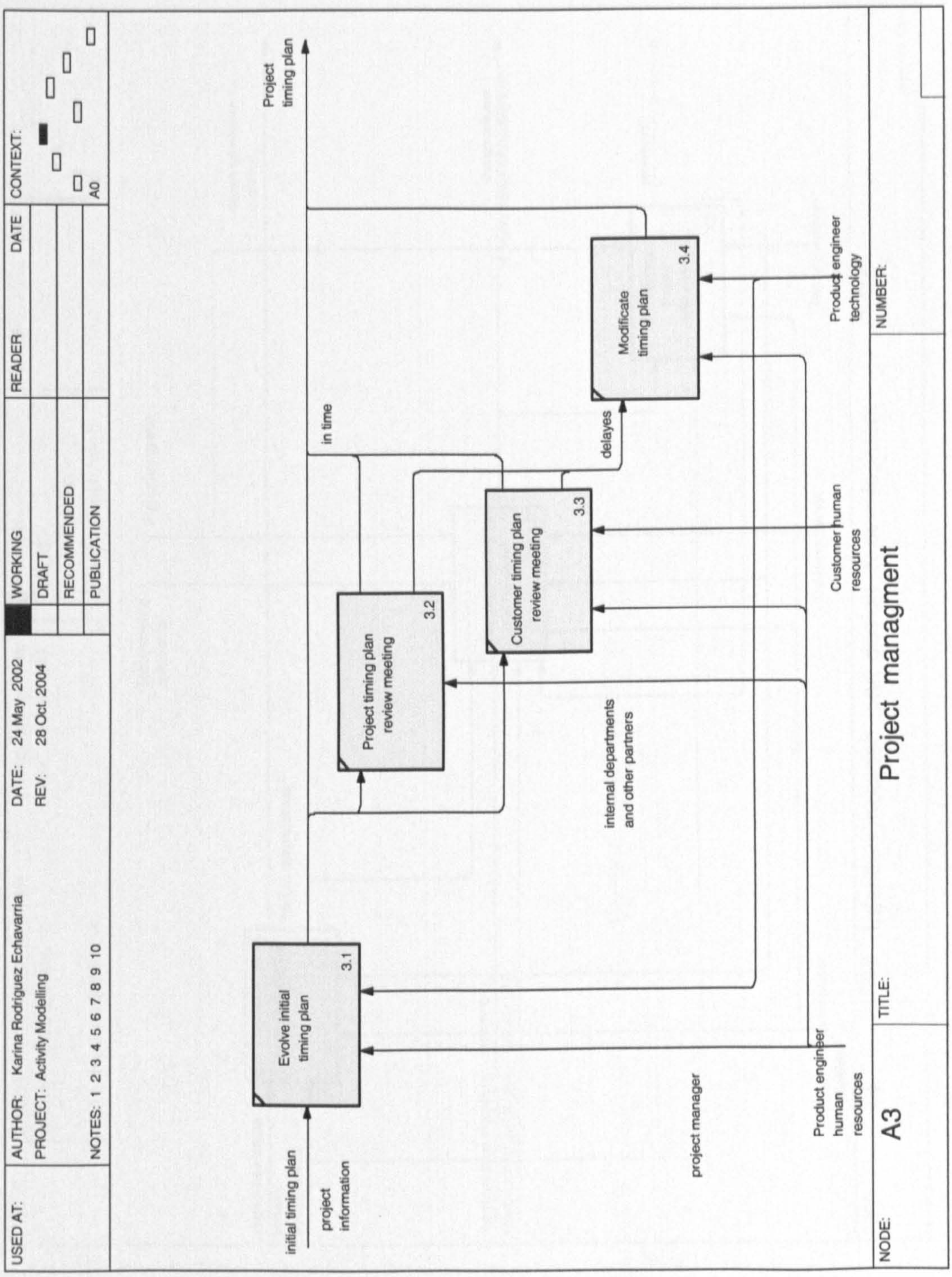


Figure 5.5 Project management activity

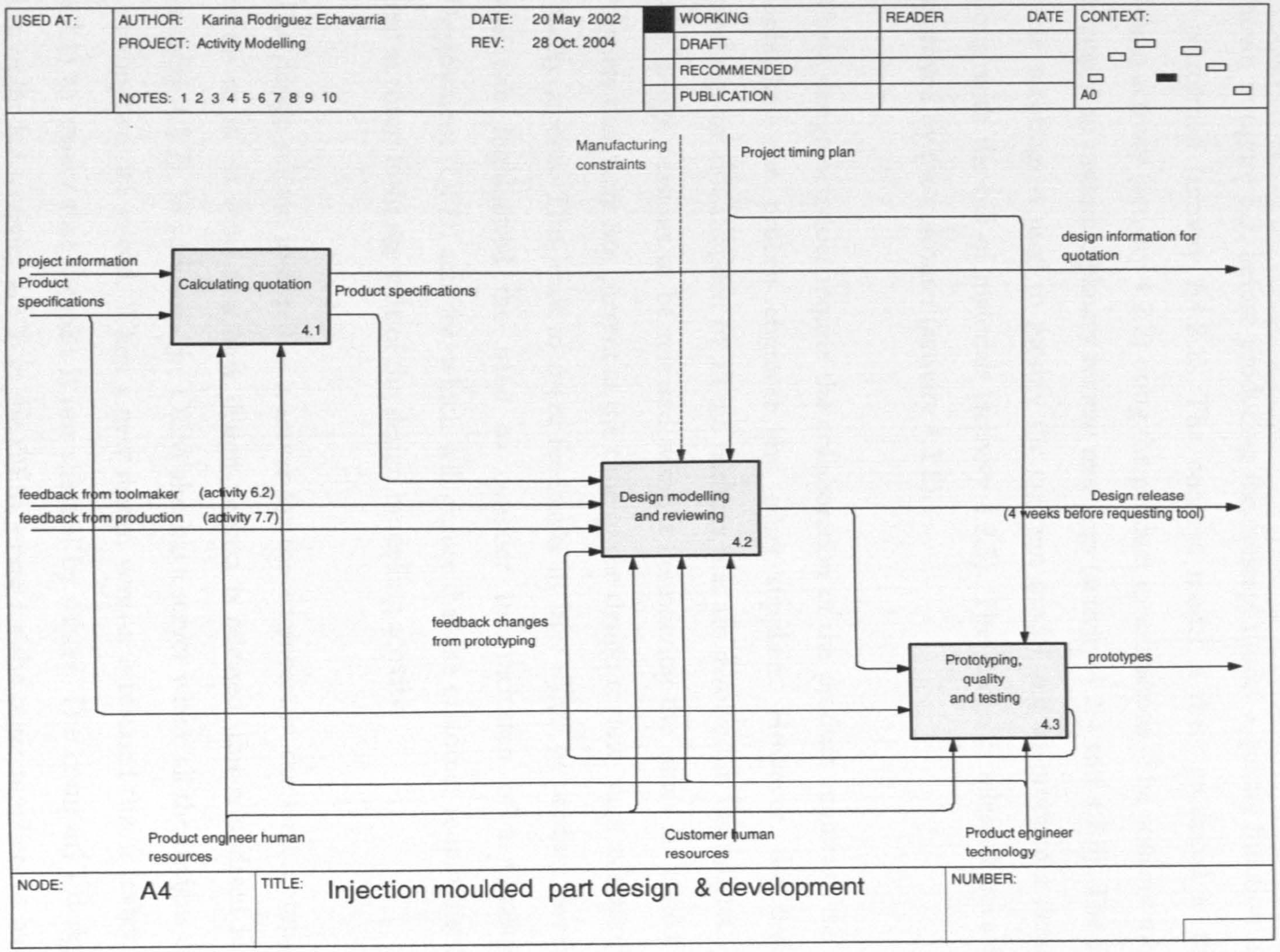


Figure 5.6 Design and development activity

5.3.4.2 Design modelling and reviewing activity

Once the customer has accepted the quotation, the design modelling activity begins. As shown in figure 5.7, before producing the concept model a quality function deployment is performed (activity A4.2.1). The concept model is then produced in the concept design activity (activity 4.2.2) using the product specifications. The concept model is then discussed in multidisciplinary review meetings (activity 4.2.4 and 4.2.6). The feedback of these meetings is used to modify the concept model and to produce a detailed design along with the bill of materials (activity 4.2.5). The design is released once it has been approved by the customer (activity 4.2.8).

These design activities require the collaboration of the product engineers, the OEM, the toolmaker, the process engineer and other suppliers. However, the design review meetings are not attended by all the partners that are involved in the project. This allows new design versions to be released without considering the issues associated with those partners that were not present at the time of the design review, such as manufacturing or quality aspects. This leads to more feedbacks in later stages of product development. It was also highlighted the need to consider the inclusion of a Quality Function Deployment (QFD) activity, which will ensure that the customer requirements are taken into account from the start of the design modelling activity.

The design release is stored in a server for the engineers in the same company to have access to it. As soon as a new design version is released, this is updated in the server (activity 4.2.7). In addition, the OEM also has a server where all the designs of other first tier suppliers are stored. When a new design version is released this is reviewed (activity 4.2.1) to ensure that the part is not affected by others. The company's design release is also uploaded (activity 4.2.3) in the OEM server for the other suppliers to have access to it.

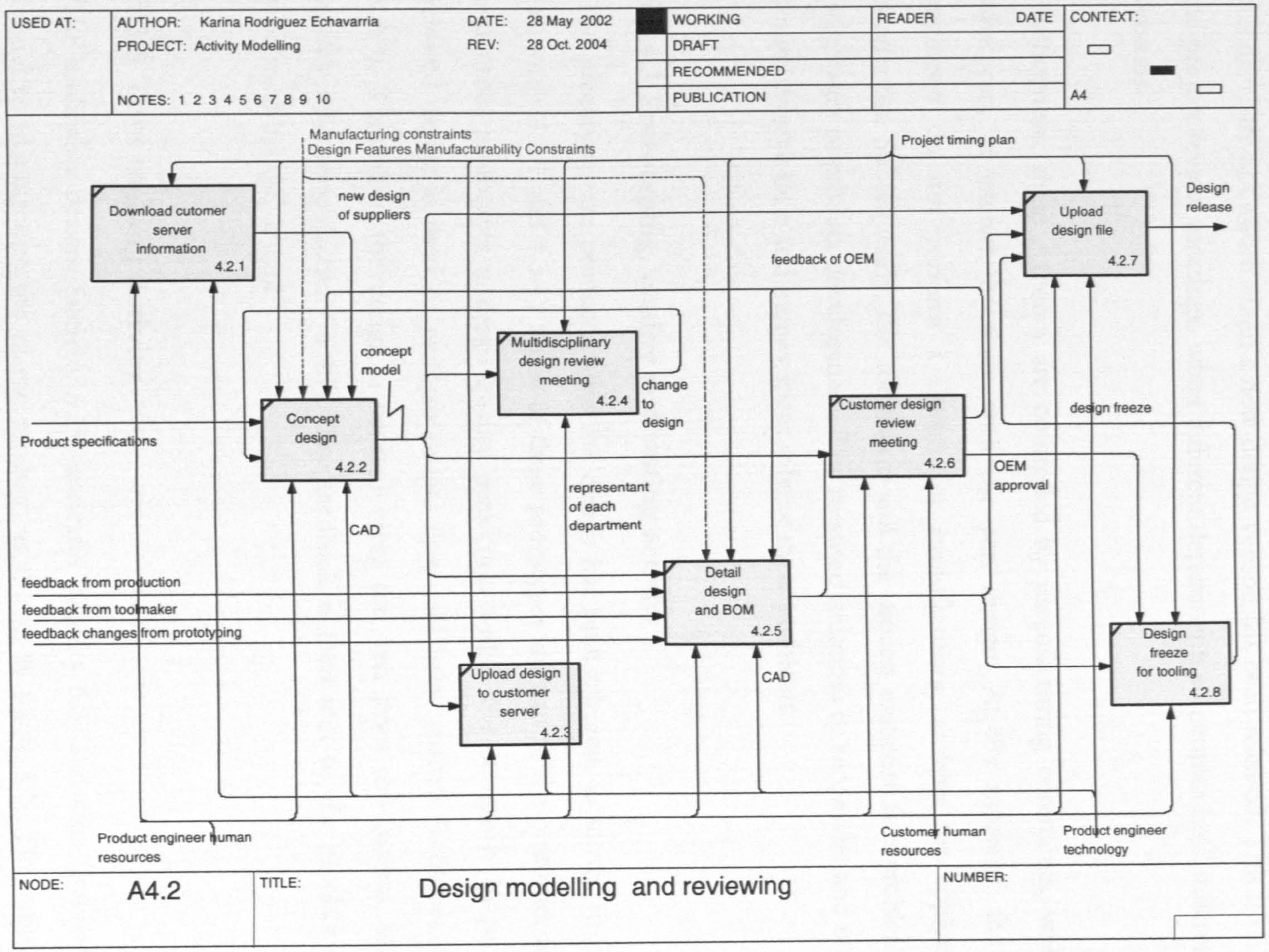


Figure 5.7 Design modelling and reviewing activity

One of the problems identified during these activities is that despite having a shared repository, the collaborators (i.e. departments inside the company, toolmaker and process engineer) are not aware when a new design version has been released. This is reflected in the project review meetings, when different departments sometimes hold different design versions.

Furthermore, these activities are controlled by manufacturing constraints, which limit what can, or cannot, be done with the part design. At the moment, the product engineers do not perform a design for manufacturing analysis for applying these constraints. Subsequently, the toolmaker and the process engineer face problems during later stages of product development. The required iterations to reconsider and modify the design involve time and money spent solving these problems.

5.3.4.3 Prototyping, quality and testing activity

The prototypes are produced after the design has been released, as shown in figure 5.8 (activities 4.3.1 and 4.3.4). Some of these prototypes are sent to the customer for testing and other prototypes undergo quality inspection within the company. In parallel, the released design is used to perform mould flow and stress analysis (activities 4.3.2 and 4.3.3). Thereafter, the design is verified using the data from the analysis, testing and quality inspection (activity 4.3.7). The feedback is then sent to the product engineer, who modifies the design.

5.3.5 Tool making activity

The toolmaker designs (activity 6.2), fabricates (activity 6.3) and tests (activity 6.4) the mould to manufacture the plastic product, as shown in figure 5.9. The input of these activities is the design release and feedback from the process engineer. The final result is the mould, which is sent to the process engineer to be used during production.

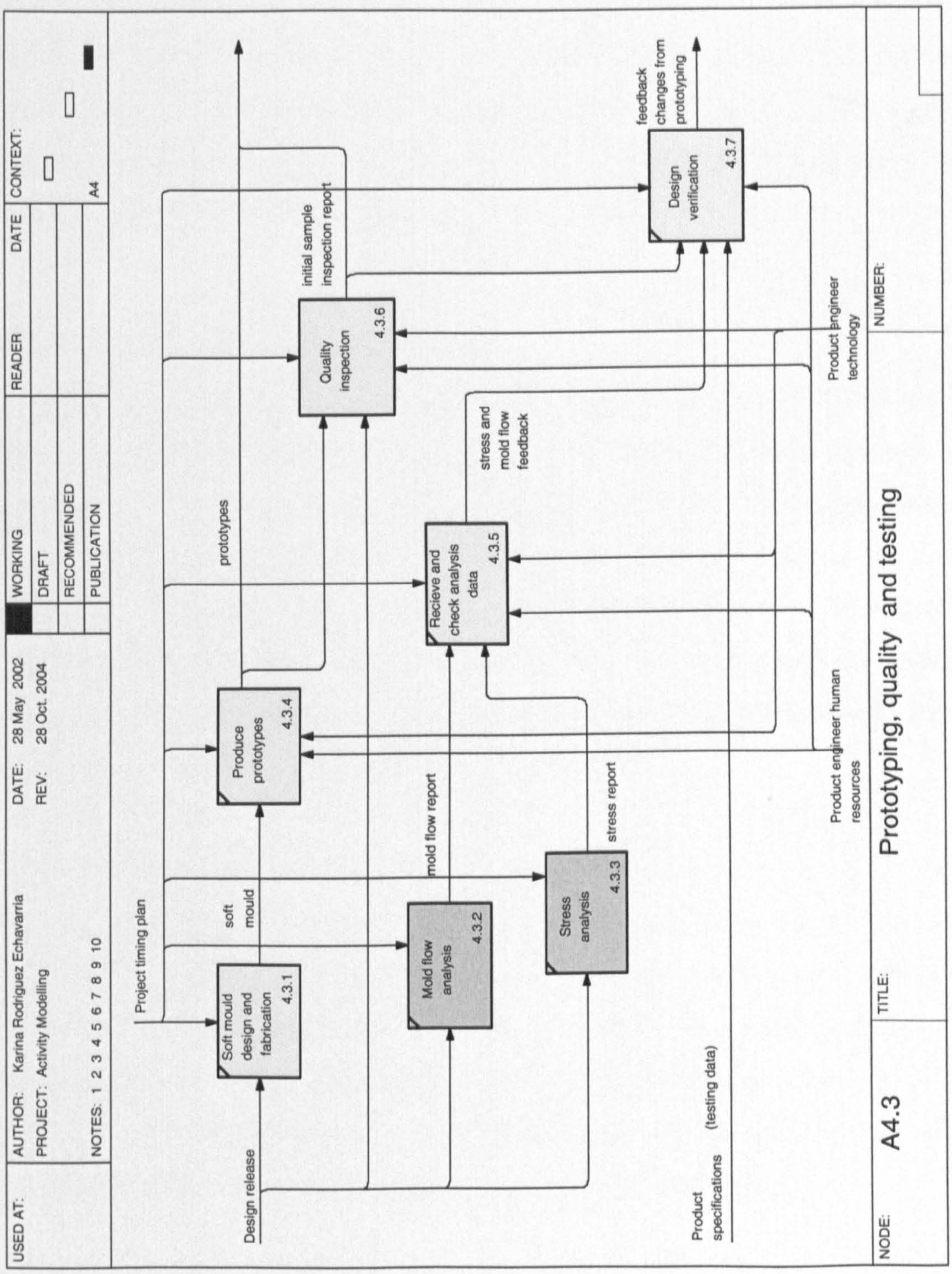


Figure 5.8 Prototyping, quality and testing activity

Manufacturing constraints limit the decisions that can be taken during the mould design and fabrication activities. If the toolmaker does not consider these constraints at the initial stage, problems will arise later. Furthermore, the engineering decisions that are taken during the mould design activity have a direct effect on the final part appearance. These issues cannot be considered during the initial design because the product engineer does not have the knowledge to take decisions related to the mould at the time. Therefore, problems arise later causing feedback between the collaborators.

5.3.6 Process engineering activities

During this activity, the process is planned, the manufacturing resources of the company are allocated and the product is manufactured. As figure 5.10 shows, process planning (activity 7.1), selection of the production equipment (activity 7.2) and process parameters (activity 7.6) are performed by the process engineer according to the project information and design release. Once the process plan and the capacity plan have been produced, the schedules for the bought out goods (activity 7.5) are sent to the third tier suppliers. The suppliers send their products during production based on this schedule, after which products are produced and shipped to the customer.

Manufacturing constraints control the decisions that can be taken during the activities of process engineering, such as selection of production equipment and process parameters. This knowledge is in the experience of the engineer and is not formally captured. Moreover, there are other problems that are caused by design decisions that were previously taken without considering manufacturing constraints. In order to overcome these problems, feedback is sent to product design and development as well as tool making activities causing more iterations during product development.

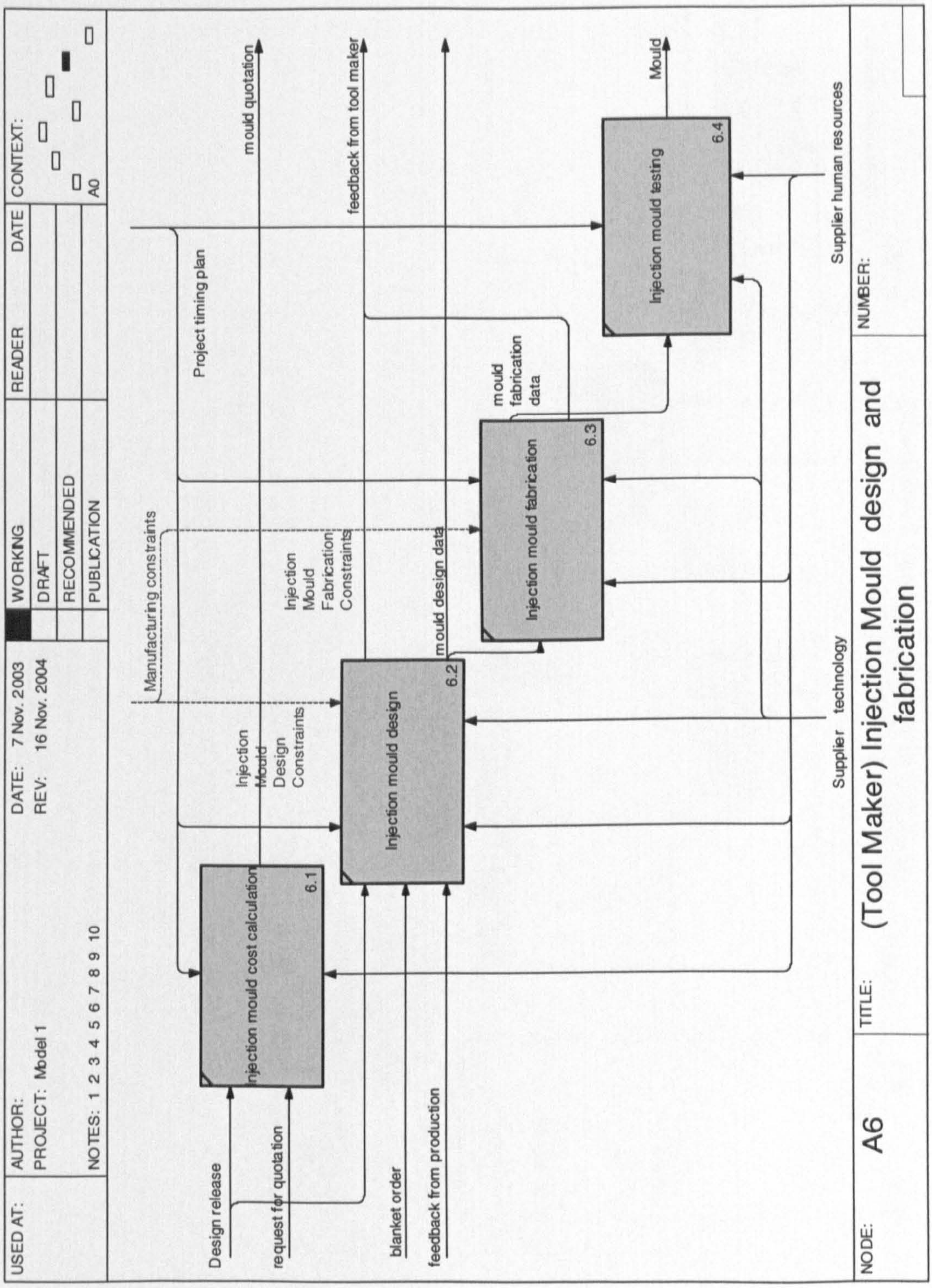


Figure 5.9 Injection mould design and fabrication activity

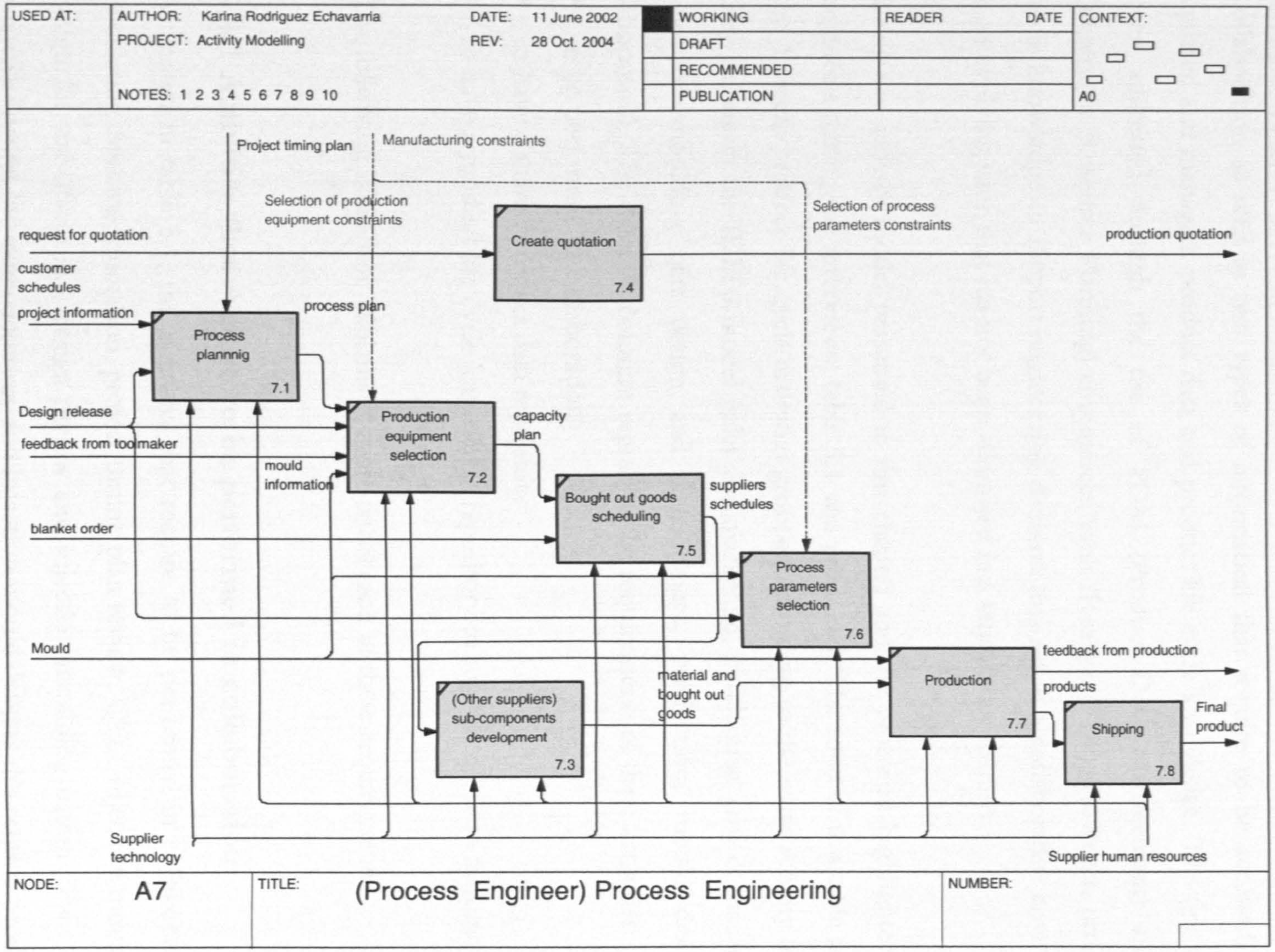


Figure 5.10 Process engineering activity

5.4 Activity modelling findings

The activity modelling led to the identification of the activities that require better collaboration as well as two types of information that require to be accessed, shared, updated and managed: product data and product life cycle knowledge. The first type has been addressed through the use of PDM (Product Data Management) and ERP (Enterprise Resource Planning) commercial tools (Rezayat 2000). However, product life cycle knowledge to support engineering decision making in a collaborative environment is an evolving issue that has not been addressed in a satisfactory manner.

Using the activity model presented in this chapter and the problems highlighted by the engineers during the interviews, table 5.1 was produced. The rows of this table represent the different product life cycle activities grouped according to the main activity to which they belong in the IDEF0 model (sales and marketing, purchasing, project management, injection moulding part design and development, production, mould design and fabrication). The different columns represent the requirements of these activities:

- to be performed in collaboration
- to have a shared product data repository
- to have a product life cycle knowledge repository to support decision making.

The following subsection explains in more detail each of these requirements.

5.4.1 Activities that require to be performed in collaboration

As shown in table 5.1, the activities that require to be performed in collaboration are: calculating customer quotation, project timing plan review, QFD, injection moulded part design for manufacturing, design review and injection moulding design. Two of these activities (design for manufacturing, and injection mould design) also require product data and knowledge provision, which makes them critical because the decisions taken during these activities influence, to a great extent, all other activities. Therefore, they not only require the collaboration of the distributed team members but also the provision of

knowledge in the right time, place and format in order to develop the product correctly, eliminating feedbacks and iterations.

Table 5.1 Collaboration, information and knowledge requirements identified during the activity modelling

	Activities	Performed in collaboration	Product data shared repository	PLC knowledge to support engineering decision making
Sales and Marketing	Calculating customer quotation	*	Product specifications	
Purchasing	Requesting and receiving supplier quotations		Product specifications	
Project Management	Project timing plan review	*	Project schedule	
Injection moulding part design and development	QFD	*	Customer requirements	
	Design for manufacturing	*	Design data, plastic data	Design features manufacturability constraints
	Design version control		Design data	
	Design review	*	Design data	
Production	Selection of production equipment		Design, mould data, company resources	Production equipment selection constraints
	Selection of process parameters		Design, mould data, standard mould parts data	Process parameters selection constraints
Mould design and fabrication	Injection mould design	*	Design, resources data	Injection mould design constraints
	Injection mould fabrication		Design, resources data	Injection mould fabrication constraints

5.4.2 Activities that require the sharing of data

All the activities presented in table 5.1 require different product data to be shared. For example: product specifications, project schedule, customer requirements, design data, plastic data, mould data, company resources and standard mould parts data. Other data such as plastic material, resources, and standard mould part data, is also required to support decision making and, in this research, is referred to as engineering data.

5.4.3 Activities that required to be supported by providing product life cycle knowledge

An initial insight of the knowledge required to support product development was gained during the activity modelling. This is the knowledge that affects decision making during the activities: design for manufacturing, selection of production equipment, selection of process parameters, injection mould design and fabrication. Currently, the knowledge required to support these activities is distributed among the collaborators. Moreover, the knowledge is not captured in a standardised format but it resides in books, reports, other literature and in the experience of the engineers. This becomes a problem because the engineers need to be present during collaboration otherwise the knowledge is not available. In addition, the knowledge depart with the engineers when they cease working for the company.

As shown in table 5.1, an abstract view of the required knowledge that constrains the key activities of product development was identified in the activity model as below:

- Design features manufacturability knowledge, constrains the engineering decisions made during the consideration of manufacturability issues on the part.
- Mould design knowledge.
- Mould fabrication knowledge.
- Selection of production equipment knowledge.
- Selection of process parameters knowledge.

This identification highlighted the geographically distributed nature of the knowledge. In this research, the geographically distributed knowledge is brought together by the Manufacturing Knowledge Model. The KdCPD system proposed in this research is based on this model to support decision making during the key activities identified in this chapter. Such a KdCPD system architecture is presented in detail in the following chapter.

5.5 Closing Remarks

This chapter presented the activity modelling of collaborative product development among an extended enterprise. IDEF0 was used as modelling technique to identify the information and knowledge driven manufacturing activities. The findings of this model were used as a basis to propose the KdCPD system architecture which is presented in the following chapter. In addition, the model was used to develop the Manufacturing Knowledge Model with the knowledge that requires to be provided during the identified activities.

Chapter 6

Knowledge Driven Collaborative Product Development System Architecture

6.1 Introduction

This chapter presents the proposed KdCPD system architecture, which addresses the research issues (see section 4.2) and requirements identified during the activity modeling (see section 5.4). Section 6.2 provides an overview of such architecture, whereas section 6.3, 6.4 and 6.5 describe in detail each of its elements.

6.2 KdCPD system architecture description

In order to give a clear description of the system architecture it is necessary to subdivide the architecture in parts, called layers. As shown in figure 6.1, the architecture contains three layers: information layer, application layer and end user layer or client. This organisation allows the physical separation of information, knowledge and applications, which should be distributed among the different companies of an extended enterprise. In such a system, the end user layer is situated in the user's desktop and is connected to the application server (application layer), which in turn is connected to the information and knowledge databases (information layer). The following sections will describe each of these layers in more detail.

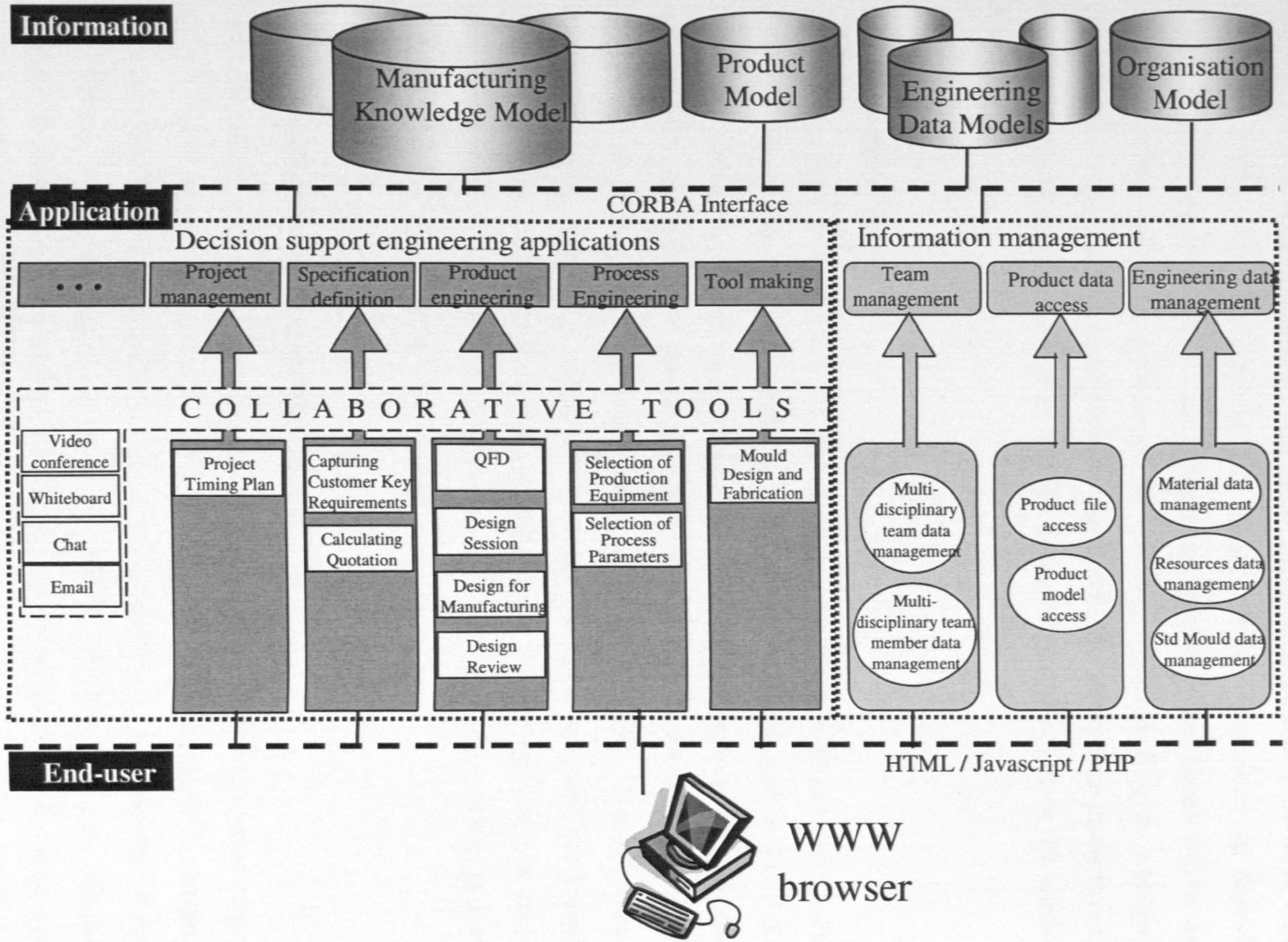


Figure 6.1 Knowledge driven Collaborative Product Development system architecture

6.3 Information layer of the KdCPD system architecture

The information layer contains databases with the information and knowledge required to support CPD. This layer contains: the Product Model, Manufacturing Knowledge Model, Engineering Data Models and Organisation Model. These models will be further explained in the following subsections. In addition, CORBA (Object Management Group 2003) is used as the communication protocol for the transparent integration of the information and knowledge databases with the engineering applications in the application layer.

6.3.1 Product Model

As has been argued in section 4.2.2, the inclusion of a Product Model that covers the complete product life cycle is an evolving research issue. Standards such as STEP (Owen 1994) have proven to provide a good solution for capturing product life cycle data. In this research, the structure of the Product Model is not fully based on this standard but instead uses a feature based approach (Latif and Hannam 1996). This approach was chosen because it enables the integration of product life cycle data with the knowledge required to support a range of engineering applications. This integration was critical in order to capture, represent and provide product life cycle knowledge, which, as discussed in section 4.2.3, is an emerging research issue.

6.3.2 Manufacturing Knowledge Model

Decision making during collaborative product development is difficult because companies do not have access to knowledge and information of their distributed partners. To overcome this issue, it is necessary to have a distributed source of knowledge to support the different engineering activities. In this research, the Manufacturing Knowledge Model is an information model that captures manufacturing constraints that limit decision making during CPD. This model and the methodology followed for its development are original contributions of this research.

The Manufacturing Knowledge Model is the source of knowledge required to support the decision making during the engineering applications presented in the application layer section. In addition, the impact of one engineering decision on other applications is highlighted due to the interaction between the knowledge captured in the model. The development of this model is presented in detail in chapter 7.

6.3.3 Engineering Data Models

The Engineering Data Models capture plastic, resource and standard mould data. This data is required to support different engineering activities as mentioned in section 5.4.2.

6.3.4 Organisation Model

In order to provide project and team management applications, the Organisation Model captures data related to a project, its tasks and the multidisciplinary team. Other security issues, such as user passwords and access rights are also considered.

6.4 Application layer of the KdCPD system architecture

The application layer consists of two elements: decision support engineering applications and information management tools. These elements are discussed in the following sections.

6.4.1 Decision support engineering applications

In response to the requirements identified during the field study and activity modelling, this layer provides a range of applications to support key product life cycle activities that need to be preformed in a collaborative manner throughout CPD. This research is concerned with the injection moulded product development; hence the proposed decision support engineering applications are project management, specification definition, product engineering, process engineering and tool making. Each of these applications contains subapplications in order to provide support for specific engineering

activities. The supported engineering activities could be extended in the future by adding more applications.

In order to address the requirement of knowledge provision, the engineering applications support decision making through feedback advice based on relevant product life cycle knowledge. In addition, communication tools are provided to support the collaboration of the geographically distributed team members.

As a result of the interaction between product data and product life cycle knowledge, the engineering applications provide the following advantages:

- A level of automation in the decision making process.
- The capability to be performed in parallel.
- The flexibility to move from one application to another without the need to follow a rigid sequence (assuming there is a certain level of product data available).

The following subsections will describe in more detail the decision support engineering applications.

6.4.1.1 Project management applications

The industrial field study identified the requirement to support collaboration during project management. Therefore, this application uses the principles and fundamentals of project management techniques (Project Management Institute 2000) to provide the involved team members with a project timing plan, which includes tasks status, times and required resources. This plan could be shared amongst the team within a virtual meeting environment using the communication tools.

6.4.1.2 Specification definition applications

Customer requirements need to be captured to ensure that the voice of the customer is represented throughout product development. These requirements, which are referred as

product specifications, are captured, stored and accessed from the Product Model through the Capturing Customer Key Requirements application. Furthermore, the Calculating Quotation application provides an environment where the different departments and suppliers collaborate to produce the quotation of the project.

6.4.1.3 Product engineering applications

The product engineering applications consist of several subapplications: Quality Function Deployment (QFD), Design Session, Design for Manufacturing, and Design Review. These subapplications could be performed in a collaborative manner, as it was highlighted in the field study. A description of each of them is given below.

Customer requirements are needed to perform Quality Function Deployment (QFD) in a collaborative environment. These requirements are captured during the Capturing Customer Key Requirements application. Moreover, the communication tools are available to facilitate the collaboration between the product engineering company and the customer when developing the QFD for a plastic part.

This architecture assumes that the conceptual design has already been agreed. For this, internet based applications are available, as explained in the literature survey (see section 2.3.5). Thereafter, during the Design Session application the engineer defines the plastic part in terms of features, such as wall, ribs and webs. The data of these features is stored in the Product Model and used by different engineering applications to support decision making after invoking the required knowledge from the Manufacturing Knowledge Model. The geometric representation of the part is available in a 3D viewer.

The Design for Manufacturing (DFM) application ensures that the functional features of the plastic part are designed within manufacturing constraints. These constraints are accessed from the Manufacturing Knowledge Model, and feedback advice is provided in case a feature falls beyond limitations. These manufacturing constraints are explained in section 7.4.1. In addition, the application supports the location of gates and ejection pins

in the part in order to avoid weld lines. This application could be concurrently accessed by the geographically distributed team members to simulate a collaborative design session. Therefore, the inclusion of this application addresses the need to perform design for manufacturing during the design modelling and reviewing activity, as explained in section 5.3.4.2.

The Design Review application provides a collaborative environment, where the geographically distributed team members discuss and modify the design using several communication tools.

6.4.1.4 Process engineering applications

The process engineering applications consist of two subapplications: Selection of Production Equipment and Selection of Process Parameters.

The Selection of Production Equipment application supports the selection of a suitable injection moulding machine for the production of a specific plastic part. For this, it is necessary to consider the part design data, available from the Product Model, and the company's resources data, available in the Engineering Data Models. Furthermore, the knowledge required for such selection is available from the Manufacturing Knowledge Model. This knowledge is explained in section 7.4.2.

The Selection of Process Parameters application provides advice regarding the optimum operation parameters (i.e. the injection pressure, the plastic material melting temperature, the mould temperature and the cycle time) for the production of a specific plastic part. This advice is based on knowledge captured in the Manufacturing Knowledge Model explained in section 7.4.3, as well as on part design and mould design data available from the Product Model.

6.4.1.5 Tool making applications

The Mould Design and Fabrication application supports the design and fabrication of the different injection mould elements, such as: core, cavity, feed system (sprue, gate and runner), venting system, cooling system and ejection system. For this, it is necessary to consider the part design data, available from the Product Model, and the standard mould data, available from the Engineering Data Models. Furthermore, the knowledge required to support the automation of the mould design and fabrication activity is accessed from the Manufacturing Knowledge Model. This knowledge is explained in section 7.4.4 and 7.4.5. Furthermore, the mould design could be performed during a collaborative session between the geographically distributed team members.

6.4.2 Information management applications

The proposed KdCPD system is based on timely and accurate provision of information, which in turn supports a number of decision support engineering applications. Hence, the information management applications are provided in order to: 1) manage the geographically distributed collaborative team, 2) control information access and 3) manage engineering data. These applications are: team management, product data access and engineering data management. These are discussed in the following subsections.

6.4.2.1 Team management applications

The geographically distributed team, which in this research is referred to as virtual team, needs to be formed prior to the start of product development. Thereafter, the team members' information, responsibilities and access rights are controlled and managed through these applications.

6.4.2.2 Product data access application

Different product related documents and files could be shared among the virtual team through the KdCPD system. To support this, the Product File Access application is provided to upload/download these documents. In addition, the Product Model Access

application provides an interface to view the plastic part, mould and injection machine data stored in the Product Model. Only assigned end users are allowed to perform these activities.

6.4.2.3 Engineering data management application

The management and maintenance of engineering data, such as resources, material and standard mould parts, is performed by the CPD system administrator. For this, interfaces to store and update engineering data are available for the authorised users.

6.5 End user layer (client)

The end user layer is the front end of the system. It mainly consists of a web browser, such as Internet Explorer or Netscape, to access the different decision support engineering applications and collaborative tools regardless of the physical location of the engineer. Different client side technologies, such as HTML, Javascript and PHP, are used to develop the web based system interface.

6.6 Closing Remarks

This chapter presented a detailed explanation of the proposed KdCPD system architecture. The elements that compose such architecture were carefully selected in order to fulfil each of the requirements identified in previous sections. As such, the Manufacturing Knowledge Model and the information models (Product Model, Organisation Model and Engineering Data Models) are the source of knowledge and information that support decision making through the proposed engineering applications. The activities that are supported by these applications were identified during the activity modelling presented in the previous chapter. Furthermore, the following chapter explains in detail how the knowledge and information models were developed.

Chapter 7

Collaborative Injection Moulded Product Development Knowledge and Information Modelling

7.1 Introduction

This chapter describes the activities performed by the author to model the knowledge, information and organisation aspects of CPD. These activities were performed following the guide of CIMOSA reference framework, as presented in section 4.3. They resulted in the definition of the Manufacturing Knowledge Model, Product Model, Organisation Model and Engineering Data Models, which are critical components of the KdCPD system architecture proposed in chapter 6. This is because one of the main contributions of the research is the capture, representation, and provision of product life cycle knowledge in the place, time and format required to support engineering decision making in a collaborative environment. The models were built based on the activity modelling, which provided an initial insight of the information and knowledge requirements.

The knowledge modelling process, followed by the author, to identify, capture and represent both knowledge and information are described in section 7.2. Section 7.3 presents the injection moulding process in order to set the reader into context. Furthermore, the knowledge modelling is presented in section 7.4 according to the engineering activities that require to be supported, as described in section 5.4.3. Section 7.5 describes how the different knowledge is brought together in the Manufacturing Knowledge Model. Finally, the Product Model and Organisation Model are defined in section 7.6 and 7.7.

7.2 Injection moulding knowledge and information modelling

7.2.1 Knowledge modelling process

The knowledge captured in the Manufacturing Knowledge Model refers to the technological, process, resource and material considerations that constrain the engineering decisions that could be made during the different activities of product development. In this research, the author proposes a knowledge modelling process in order to identify, capture and represent product life cycle knowledge. This process is as follows:

1. Identify and gather technological, process, resources and material constraints. Firstly, the knowledge was extracted from documents, books, reports and engineer's expertise (Pye 1989; Ticona 2000; Ticona and Tim Spahr 2000; Dow 2001; Menges, Michaeli et al. 2001; Osswald, Turng et al. 2002). The constraints were then classified in categories in accordance with the activities that require to be supported as identified during the field study and the activity modelling. These categories are: design feature constraints, mould design and fabrication constraints, as well as selection of production equipment constraints and process parameters constraints. Furthermore, the constraints were subcategorised according to the individual element on which the constraints are imposed. For example, manufacturability constraints imposed on a wall feature or design constraints imposed on the gating system of the injection mould. Engineering data, such as plastic materials, resources and standard mould plates, was also identified during this stage. This data was used to build the Engineering Data Models.

2. Capture and standardise the manufacturing constraint in the form of rules and mathematic formulas. Two types of rules were captured:

- Recommendation rule: This suggests suitable value(s).
- Limitation rule: This checks that the actual value of the product data, which is being used for the current product development, is within limitations. This value refers to features dimensions, injection machines size, type of mould elements, or any product life cycle data on which the constraints are imposed.

The rules and formulas were captured in a semi structured format. This means that even when the rules can be directly mapped to a programming language, they are still using natural language in order to be understandable to the reader who is inexperienced in more structured programming languages.

3. Represent the manufacturing constraints formally using UML object oriented language[1] (Object Management Group 2003). UML was selected because this language can represent the manufacturing constraints in a suitable level of abstraction. Objects contain attributes, which describe the object represented, and methods, which describe its behaviour (Kifer, Lausen et al. 1996). As such, a manufacturing constraint was represented as an object oriented class.

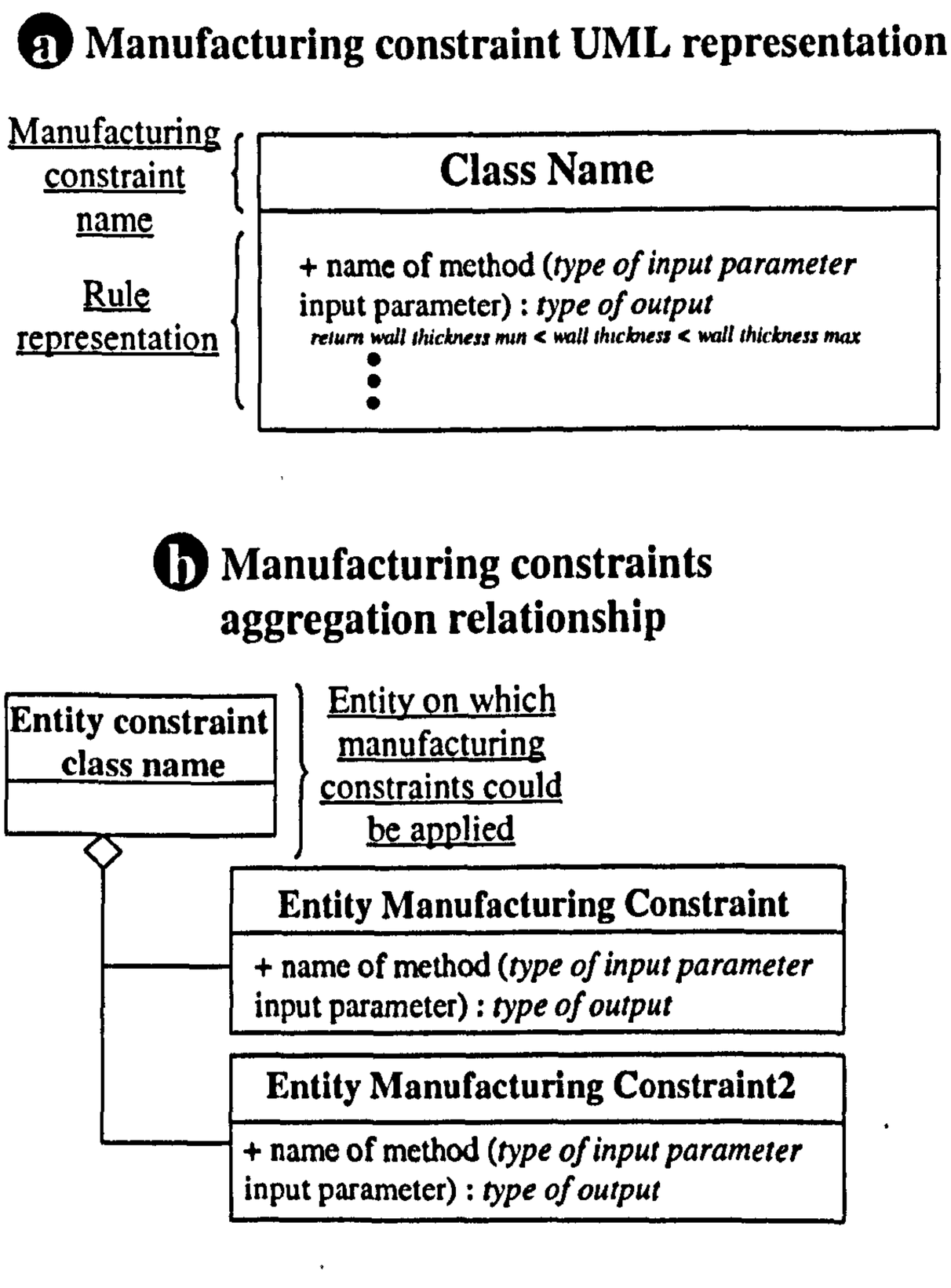


Figure 7.1 UML notation for the representation of a manufacturing constraint

[1] For a detail description of the UML language refer to appendix D.

As figure 7.1-a shows, the UML notation of a class consists of a box with two sections: the upper part contains the title of the class and the lower part contains the set of methods, which represent the rules captured in the previous step. The methods usually have a name such as "suggestValue", or "checkValue" to indicate if they produce as an output an advice or they check the actual value of an entity. Following this, the type and name of the input parameter(s) is/are specified between parentheses. The values of these parameters would be retrieved in real time from the Product Model or the Engineering Data Models. Thereafter, the type of output is specified. This type could be a number (float, double), a text (string) or a true-false value (boolean). Finally, the rule, which was captured in the previous step, is included under the name of the method. Each class can have one or more methods depending on the number of rules captured.

Furthermore, the relationships, between the manufacturing constraints and the entity on which they are applied, were represented using aggregation relationships. Figure 7.1-b shows this type of relationship. As shown in the figure, the manufacturing constraints applied on an entity are aggregated to a class named after the entity. For example, the manufacturing constraints for the wall are aggregated to the "Wall Constraints" class. By doing this, the manufacturing constraints are structured into a taxonomy. Additional notations, such as package and location notations were also used to organise the manufacturing constraints according to the supported engineering activities and the location where the knowledge is to be placed. This can be seen in the top level of the Manufacturing Knowledge Model, which is presented in this chapter in figure 7.49.

Another issue that is addressed during this step of the knowledge modelling process is the knowledge integrity of the model. As mentioned in section 4.2.3, this means that the impact of one manufacturing constraint on other engineering activities is highlighted. The integrity of the model is represented using interactions among manufacturing constraints. There are two types of interactions:

- Interactions within a same manufacturing constraint: An example of this interaction is the relationship between the gate positions and the design features of the part. As

shown in figure 7.2-a, the gate position is constrained by the position of the boss in the part.

- Interactions among manufacturing constraints: Using the same example, the gate position constraint cannot be considered without previously considering the boss manufacturability constraint. The latter knowledge is owned by the product engineering company and constrain the boss data that can or cannot be used to suggest suitable gate position (see figure 7.2-b). It is firstly necessary to ensure that the boss is within limitations because without this assurance, the advice produced by the model would be meaningless. This is because problems would still arise later, as the manufacturability of the boss has not been considered. Therefore, this interaction provides the system with an intelligent mechanism to enforce the consideration of the manufacturability of the part before giving any further advice throughout product development. Furthermore, the gate positions would have an effect on the weld line location and therefore vents would be required in these areas. This relationship is represented with an interaction between the gate position constraint and the vent position constraints (see figure 7.2.-c).

Both types of interactions are captured in the Manufacturing Knowledge Model to ensure that all product life cycle knowledge is considered at every stage of product development even if the knowledge does not reside inside a particular company.

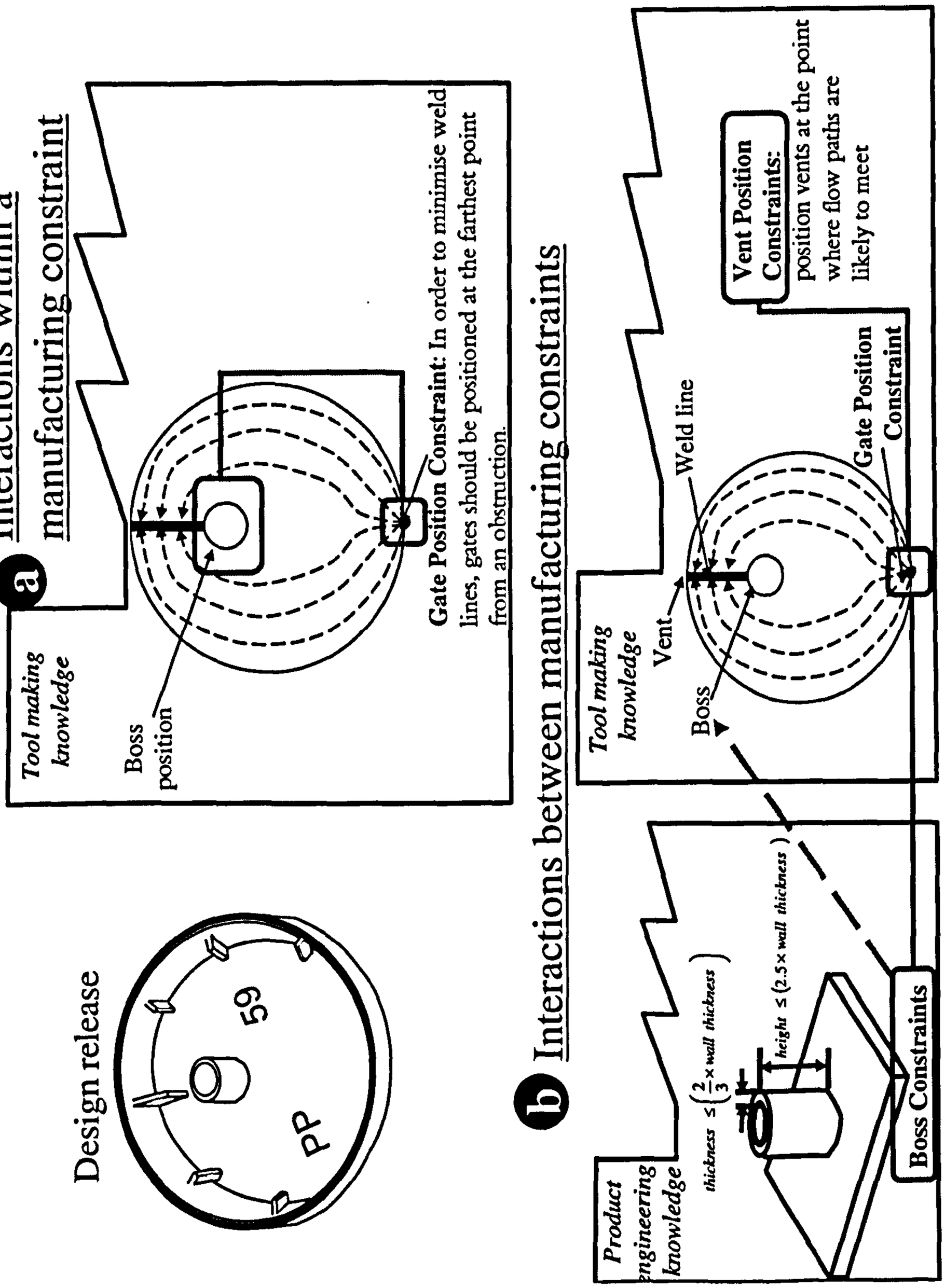


Figure 7.2 Types of interactions within the Manufacturing Knowledge Model

7.2.2 Information and organisation modelling process

The Product Model and Organisation Model were built using an object oriented approach in order to capture the product life cycle data that was relevant for the engineering applications. Hence, the Product Model uses a feature based approach, as this facilitates the knowledge provision capability of the KdCPD system. The process of information modelling included information identification and formal representation. They were conducted as follows:

1. Identify objects related to the product, such as part, wall, rib, injection mould and ejection system.
2. Formally represent each object and its interactions using UML (Object Management Group 2003) object oriented language. In this representation a class was used to represent an object and its attributes, such as length, width or position. Furthermore, the interactions between the classes were represented. For example, one part could have one or more features.

Prior to describing the identification, capture and representation of the product life cycle knowledge, next section presents an overview of the injection moulding process to set the reader into context.

7.3 Injection moulding process

7.3.1 Description of injection moulding process

Injection moulding is the process in which plastic resin is fed into the hopper of an injection moulding machine, as shown in figure 7.3. Then, they fall into a screw channel, which feeds the pellet forward inside the heated cylinder. As the mass of plastic moves towards the front of the cylinder, it is melted. The screw is allowed to travel back until sufficient quantity of molten plastic accumulates in front of the screw to fill the cavity of the mould. The screw is then pushed forward under high pressure to force the molten plastic through the machine nozzle into the closed mould (Ticona 2000).

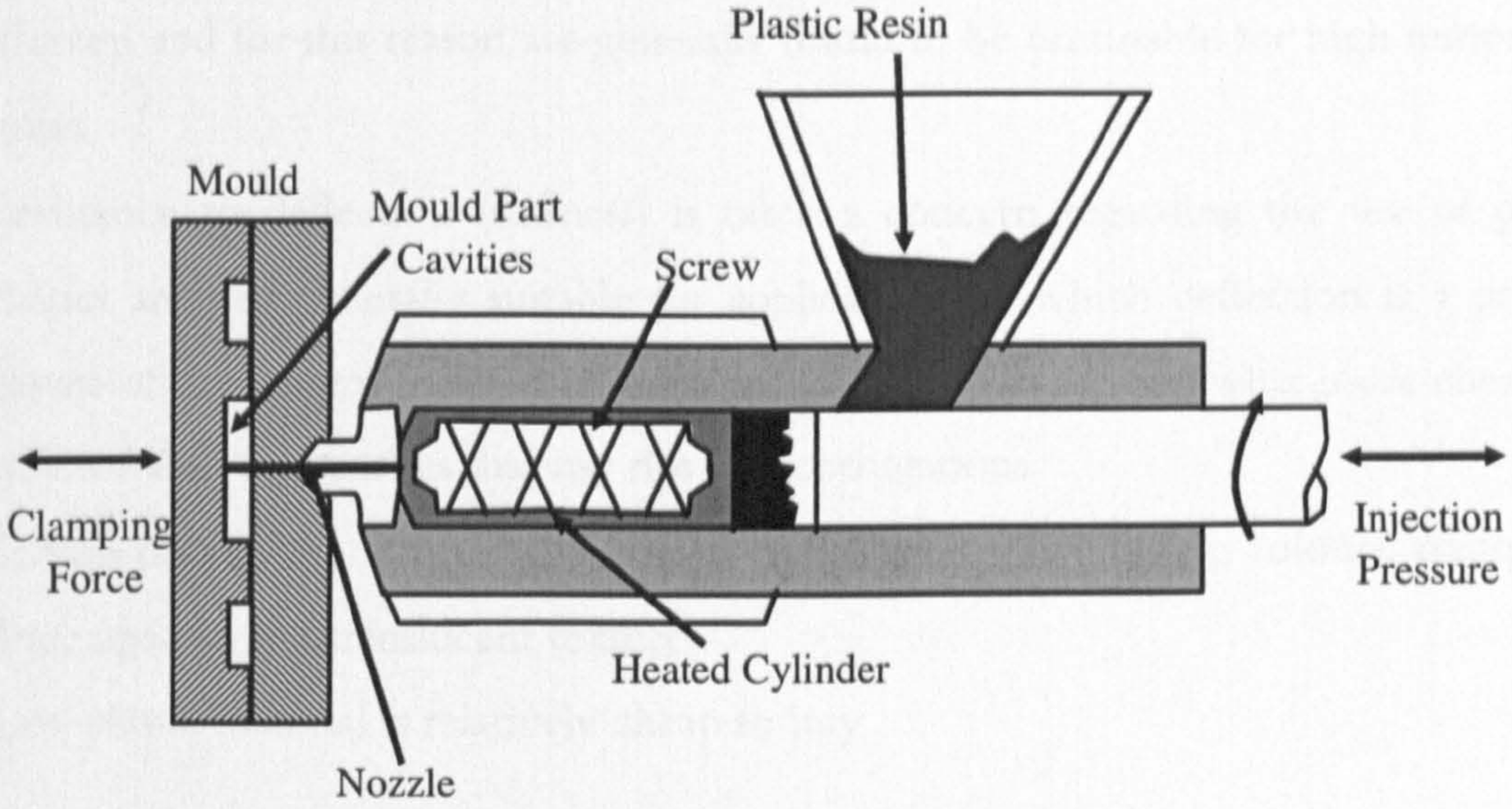


Figure 7.3 Injection moulding process

Once in the mould, the plastic flows through a distribution system called runners and then through gates into the part cavities. As soon as the plastic cools and solidifies in the mould cavity, the mould is opened and the part is removed. The key advantage of this process is its ability to accurately and repeatedly produce multifunctional or complex moulded parts in a single operation (Ticona 2000).

7.3.2 Plastic material characteristics

Plastic materials can be classified in: thermoplastics or thermosets. An essential difference between these two types is that once set, thermosets cannot be melted down and reformed. On the other hand, thermoplastics may be alternately melted, formed and remelted almost without limit (Ticona 2000).

Moreover, plastics have the following characteristics (Ticona 2000):

- Plastics are both tough and strong (for their weight) and many are practically unbreakable.
- All plastics have a coefficient of linear expansion, high electrical resistance and low thermal conductivity.

- Most plastics operating under high temperature conditions are liable to very high rates of creep and for this reason are generally found to be unsuitable for high temperature duties.
- Resistance to deflection (stiffness) is often a concern regarding the use of plastics. Plastics are not generally suitable for applications in which deflection is a principal feature of the design. Increase in stiffness may be achieved somewhat more cheaply by stiffened skin approaches that use ribs and corrugations.
- Certain plastics are available in various optical grades, including colours, transparent, clear, opaque and translucent textures.
- Raw plastic material is relatively cheap to buy.

The characteristics of the plastic material were captured in an Engineering Data Model, which is referred to as Material Model. In the model shown in figure 7.4, the plastic material is represented as an object oriented class called "Plastic" and it inherits from the "Material" class the attributes common to all materials. The attributes of the "Plastic" class represent the plastic characteristics, such as density, injection temperature, minimum and maximum wall thickness, mould temperature and shrinkage. The attributes' values were taken from different sources (Pye 1989; Ticona 2000; Menges, Michaeli et al. 2001; Osswald, Turng et al. 2002; Automation Creations 2004) and they would be used to support different engineering applications. For example, the minimum and maximum wall thickness attributes support the Design for Manufacturing application, as they are the range in which the wall is manufacturable.

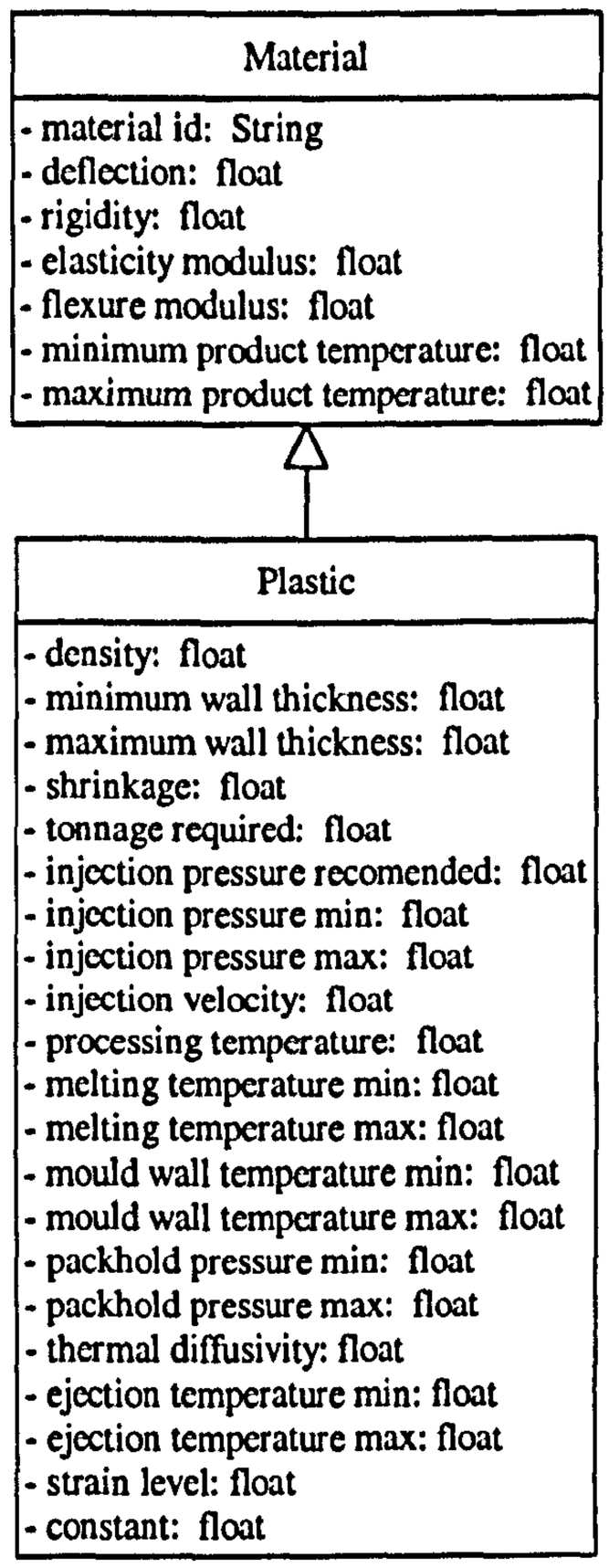


Figure 7.4 Material Model representation using UML notation

7.4 Injection moulding knowledge modelling

The following subsections present the identification, capture and representation of the injection moulding knowledge according to the following classification: design feature constraints, production equipment selection constraints, process parameters selection constraints, mould design and mould fabrication constraints.

7.4.1 Modelling of the design features manufacturability constraints

In order to represent the manufacturing constraints that affect the injection moulded part design, a feature based approach was adopted (Latif and Hannam 1996). A typical

injection moulded part has a shell type configuration with a basic surface and features that are attached to meet functional requirements. These features are referred to as moulded features as they are formed by the moulding process. By using a feature based approach, the design constraints for each of the features were identified, captured and represented. The moulded features considered were: wall, hole, reinforcement, text, snap fit, thread and corner shape. Other features such as ejection positions, gate positions, parting line and weld line were also considered.

The following subsection explains in detail the identification, capture and formal representation of the design constraints for the wall feature. Afterwards, the following subsections present the identification and capture of the design constraints for the other mouldable features. The representation of these constraints follow the same approach as for the wall and it is presented in Appendix E.

7.4.1.1 Modelling of the wall design constraints

The following subsections explain in detail the design constraints for the wall: wall thickness, wall transition and wall draft angle.

7.4.1.1.1 Wall thickness constraint

The wall is the most important feature of the injection moulded part (Dieter 2000). The knowledge required to select a suitable wall thickness was identified as follows: "Of all the issues in plastic design, selecting the proper wall thickness is probably the most important and all encompassing topic. Choosing proper wall thickness sometimes determines the ultimate success or demise of a part. A wall that is too thick would cause manufacturability problems like sink marks, shrinkage and bending; if it is too thin, on the other hand, this would cause short shot problems. As a result, the wall thickness must be within a range recommended by the material supplier" (Ticona (2000). Table 7.1 presents the range of recommended values for some of the plastic materials. These values were included in the Material Model described in the previous subsection.

Table 7.1 Ranges of recommended thickness for various plastic materials (Ticona 2000)

Plastic	Minimum wall thickness (mm)	Maximum wall thickness (mm)
ABS	0.762	3.175
Acetal	0.381	3.175
Acrylic	0.635	6.35
Nylon	0.762	3.175
Polyphenylene Sulfide	0.508	4.572
Polycarbonate	1.016	3.81
Polyester	0.635	3.175
Polyethylene (HD)	0.508	6.35
Polyethylene (LD)	0.762	6.35
Polypropylene	0.635	7.62
Styrene Acrylonitrile (SAN)	0.889	3.81

After identifying the wall thickness knowledge, this was captured in a rule based format. As explained in section 7.2, the knowledge was used for two kinds of support:

- To suggest a suitable value for the thickness of the wall.

"The thickness should be between (plastic minimum wall thickness) mm. and (plastic maximum wall thickness) mm. Recommended:

$$\frac{\textit{plastic minimum wall thickness} + \textit{plastic maximum wall thickness}}{2}\ \textit{mm.}"$$

- To check that the actual thickness of a wall thickness is within the manufacturing constraint.

plastic minimum wall thickness $\leq$ *wall thickness* $\leq$ *plastic maximum wall thickness*

Another consideration when selecting a suitable wall thickness is: "if the wall is too thick because it is subjected to any significant loading, the wall thickness must be reduced and the use of ribs should be considered (see figure 7.5)" (Ticona 2000). This knowledge was captured in the following rule:

- To recommend a rib:

if (wall thickness $\geq$ *plastic maximum wall thickness) then "Reduce the wall thickness and add a rib."*

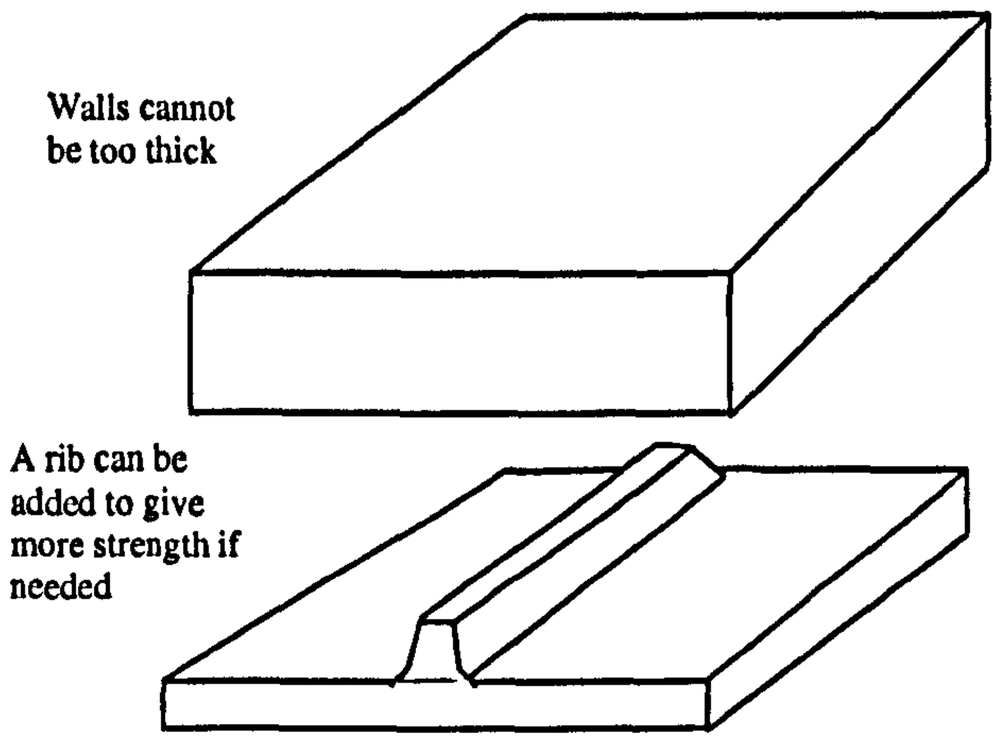


Figure 7.5 Identification of the knowledge related to the wall thickness

One final design consideration for the wall thickness is: "all the walls of the part should have the same thickness if possible. This provides an even flow of the melt during the injection" (Ticona 2000). It is important to highlight that this advice might not be taken into account if the functional purpose of the product requires differing wall thickness. The rules captured were:

- To recommend the same wall thickness than the base wall:

if (NOT first wall) and (NOT (wall thickness = first wall thickness)) then "It is recommended that the wall thickness is the same than that of the first wall: (first wall thickness)"

The rules for the wall thickness were formally represented in an object oriented class using UML notation. Figure 7.6 illustrates the "Wall Thickness Constraint" class, which methods represent the rules previously captured. The methods are:

- "suggestValue" represents the rule to suggest the suitable wall thickness.
- "checkValue" represents the rule to ensure the manufacturability of the wall.

Wall Thickness Constraints

+ suggestValue(Plastic plastic, float wall thickness, boolean first wall, float first wall thickess) : String
if (NOT first wall) and (NOT (wall thickness = first wall thickness)) then
"It is recommended that the wall thickness is the same than that of the first wall: (first wall thickness)"
else
"The thickness should be between (plastic minimum wall thickness) mm. and (plastic maximum wall thickness) mm.
Recommended: $\frac{\text{plastic minimum wall thickness + plastic maximum wall thickness}}{2}$ *mm.*
if (wall thickness = plastic maximum wall thickness) then "Reduce the wall thickness and add a rib."
+ checkValue(float wall thickness, Plastic plastic, boolean first wall, float first wall thickess) : boolean
return (plastic minimum wall thickness <= wall thickness <= plastic maximum wall thickness)

Figure 7.6 "Wall Thickness Constraint" class representation using UML notation

7.4.1.1.2 Wall transition constraint

Furthermore, another consideration that was identified is: "when wall thickness transitions cannot be avoided, the transition must be made gradually" (Ticona 2000). As illustrated in figure 7.7, the gradual transitions avoid stress concentrations and abrupt cooling differences.

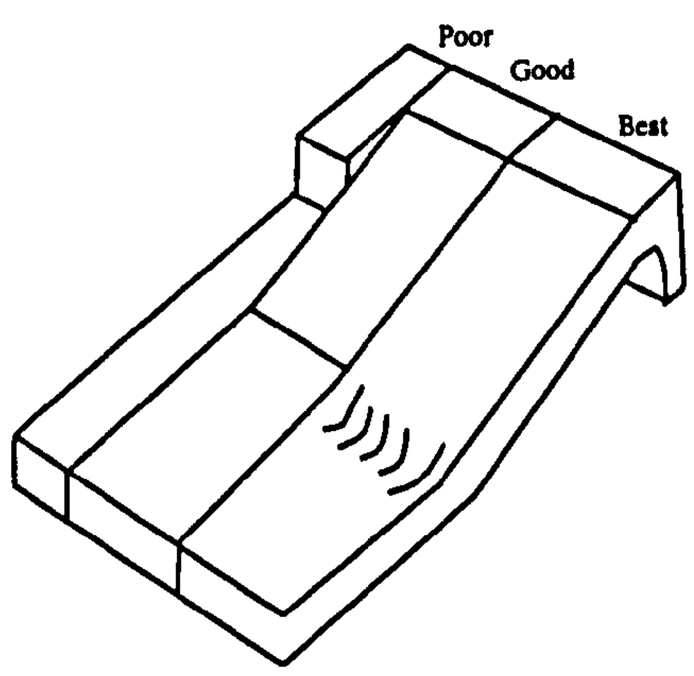


Figure 7.7 Identification of the knowledge related to a transition wall

This knowledge was captured as follows:

- To suggest a transition wall:

if (wall1 direction = wall2 direction) and (wall1 thickness ≠ wall2 thickness) then

"A connection wall is required between (wall1) and (wall2) to be used as transition."

This rule determines the need for a transition wall in case two walls are placed adjacent to one another and have different thickness. This rule assumes that the product data

representation captures the walls of a part and identifies which walls are adjacent to each other.

Wall Transition Constraint
+ suggestTransitionWall(String wall1, String wall2, float wall1 direction, float wall2 direction, float wall1 thickness, float wall2 thickness) : String *if (wall1 direction = wall2 direction) and NOT(wall1 thickness = wall2 thickness)* *"An angled wall is required between (wall1) and (wall2) to be used as transition."*

Figure 7.8 "Wall Transition Constraint" class representation using UML notation

As shown in figure 7.8, the method "SuggestTransitionWall" in the "Wall Transition Constraint" class represents the previously captured rule.

7.4.1.1.3 Wall draft angle constraint

This particular issue is of special concern to all parties involved in CPD. The identified constraint is: "a draft angle is required to all wall features that are parallel to the injection moulding machine's axis to ease the ejection of the part" (see figure 7.9). The required amount of draft depends on the surface finish of the mould. A highly polished mould requires less draft than an unpolished mould. A general rule of thumb used by product engineers and toolmakers is to use at least 1 degree of draft angle (Ticona 2000).

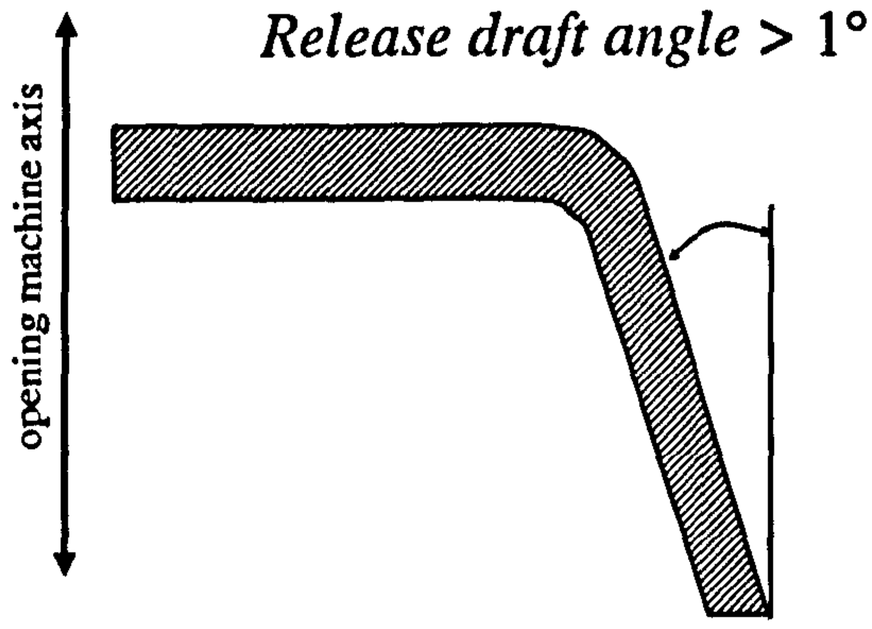


Figure 7.9 Identification of the knowledge related to the wall draft angle

This constraint was captured in the following rules:

- To suggest a draft angle:

  if (wall is parallel to machining axis) then "The draft angle should be more than 1°"

- To check that the actual draft angle of the wall is within the manufacturing constraint:

  if (wall is parallel to machining axis) then (draft angle ≥1)

These rules are represented formally in the "Wall Draft Angle Constraint" class, which is shown in figure 7.10.

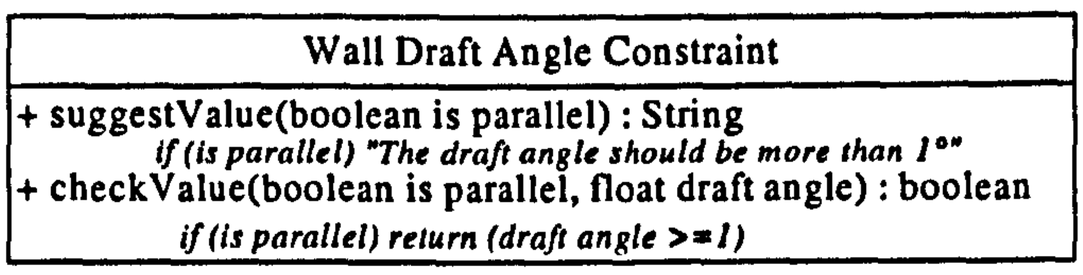


Figure 7.10 "Wall Draft Angle Constraint" class representation using UML notation

Finally, after all the manufacturing constraints of the wall were represented using UML notation, they were linked with the "Wall Constraints" class, as shown in figure 7.11. This class has an aggregation relationship with all the constraints described before.

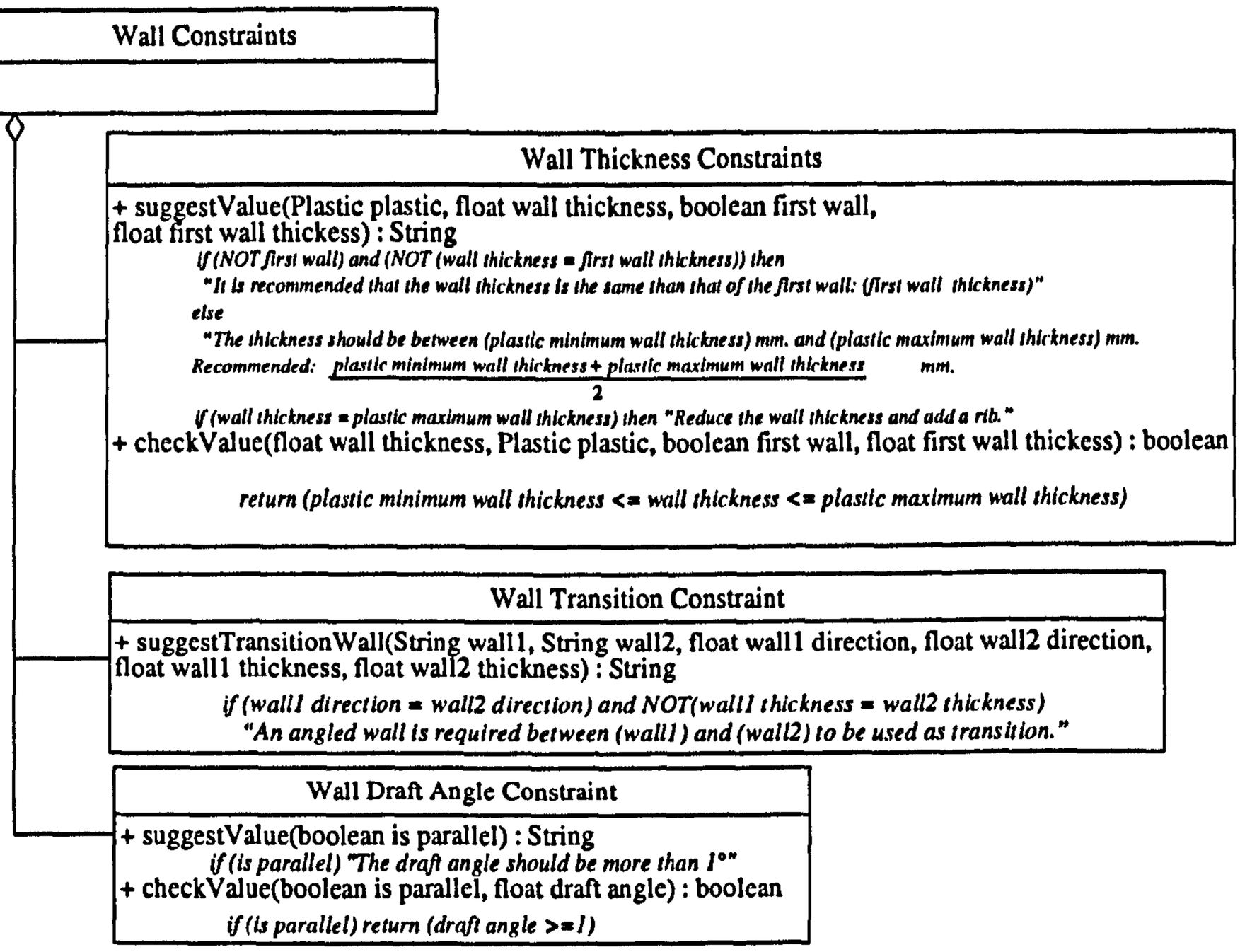


Figure 7.11 "Wall Constraints" class representation using UML notation

7.4.1.2 Modelling of the reinforcement design constraints

Reinforcement features, such as ribs, bosses and webs, are often used to give better rigidity and stiffness to the part. Their constraints are explained in the following subsections.

7.4.1.2.1 Rib constraints

Walls, in general, can be made thinner by using ribs as they provide the same rigidity to the part but with lower material cost. Although the use of ribs improves the stiffness of the part, they also cause manufacturability problems, such as sink marks, warping and appearance problems. In order to avoid this, certain guidelines must be followed (GE Plastics 1999):

- The width of a rib at the intersection with the wall must be equal or less than 2/3 the thickness of the wall (see figure 7.12-a). This value is recommended because the intersection can develop a mass of material if the rib thickness gets too big. This can

affect the fill pattern within the mould and can result in sink on the wall opposite the rib.

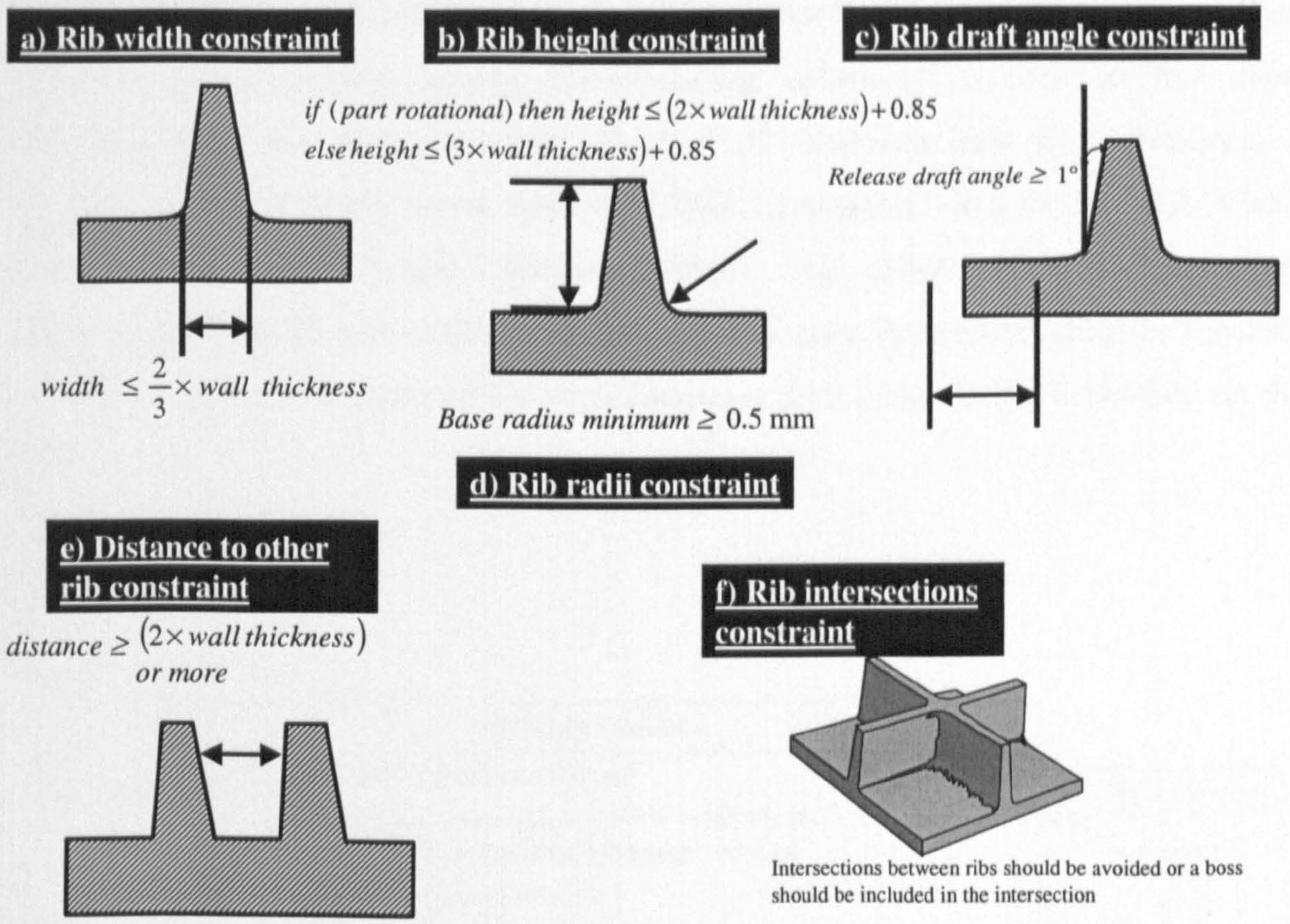


Figure 7.12 Capturing rib manufacturability constraints

- The maximum rib height must be less or equal to three times the wall thickness plus 0.85, as figure 7.12-b shows. The reason is that deep ribs become difficult to fill and they may stick in the mould during ejection. For a rib placed in a rotational part, the rib height must be less or equal to two times the wall thickness plus 0.85.
- As figure 7.12-c illustrates, a draft angle of minimum one degree is essential to ease the ejection of the part if the rib is parallel to the injection moulding machine's axis.
- The intersection at the base of the rib must radii (see figure 7.12-d). A minimum radius of 0.5 mm at the base is suggested. This radius eliminates the sharp corner and stress concentration. The material flow and cooling are also improved.

- Spacing between two parallel ribs must be a minimum of 2 times the wall thickness to keep the mould from developing cooling problems (see figure 7.12-e).

The constraints for the rib width and height depend on the wall thickness value. Therefore, an interaction among manufacturing constraints is used to link these constraints with the wall constraints. Figure 7.13 illustrates how this interaction is represented using a dotted arrow from the "Wall Constraints" class to the "Rib Width Constraint" and "Rib Height Constraint" classes. This interaction means that before considering the width and height of the rib, the wall manufacturability must be ensured. The same interaction is captured for other features if their design is also dependant on the wall.

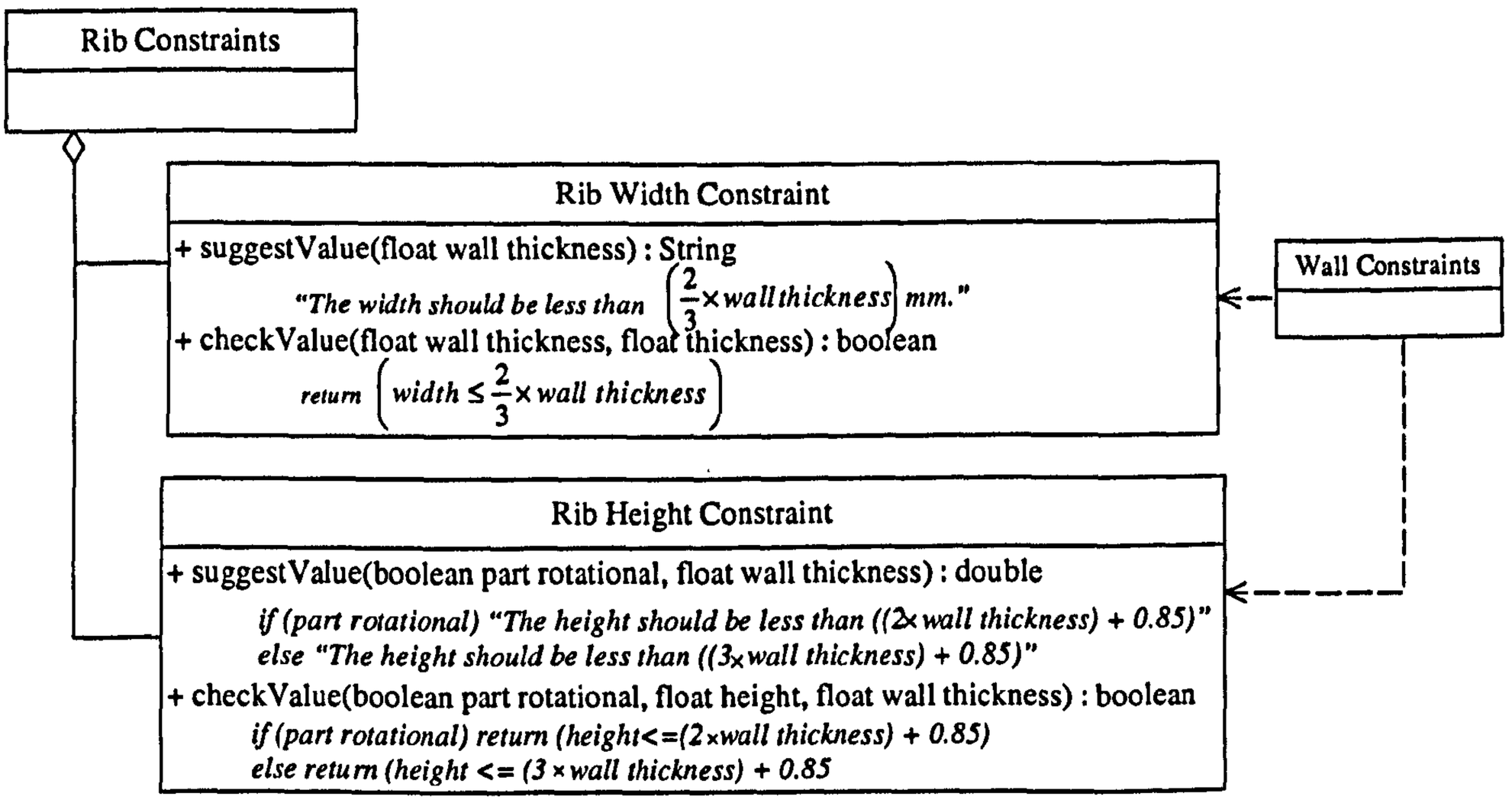


Figure 7.13 Representation of the interactions among the manufacturing constraints for the rib and the wall using UML notation

7.4.1.2.2 Boss constraints

Bosses are commonly found in injection moulded parts and serve as mounting or fastening points. As with the rib design, its constraints depend upon the wall on which they are placed (GE Plastics 1999; Ticona 2000):

- As figure 7.14-a shows, the height of the boss must remain less or equal than 2.5 times the thickness of the wall on which the boss is placed. A tall boss will generate a material mass at the base, which will require more time to cool down, thus, increasing the cycle time.
- When bosses are designed to accommodate self tapping screws, the inside diameter and wall thickness must be controlled to avoid excessive build up of stresses in the boss. For this reason, the thickness of the boss must be less or equal to 2/3 the thickness of the wall, as figure 7.14-b illustrates.
- As with the rib, the boss must also have a minimum radius of 0.5 mm at the base (see figure 7.14-c).
- A draft angle of at least 1 degree is needed to ease the release from the mould in ejection if the boss is parallel to the injection moulding machine's axis (see figure 7.14-d).
- The minimum distance between two bosses must be kept to equal or more than twice the wall thickness (see figure 7.14-e).

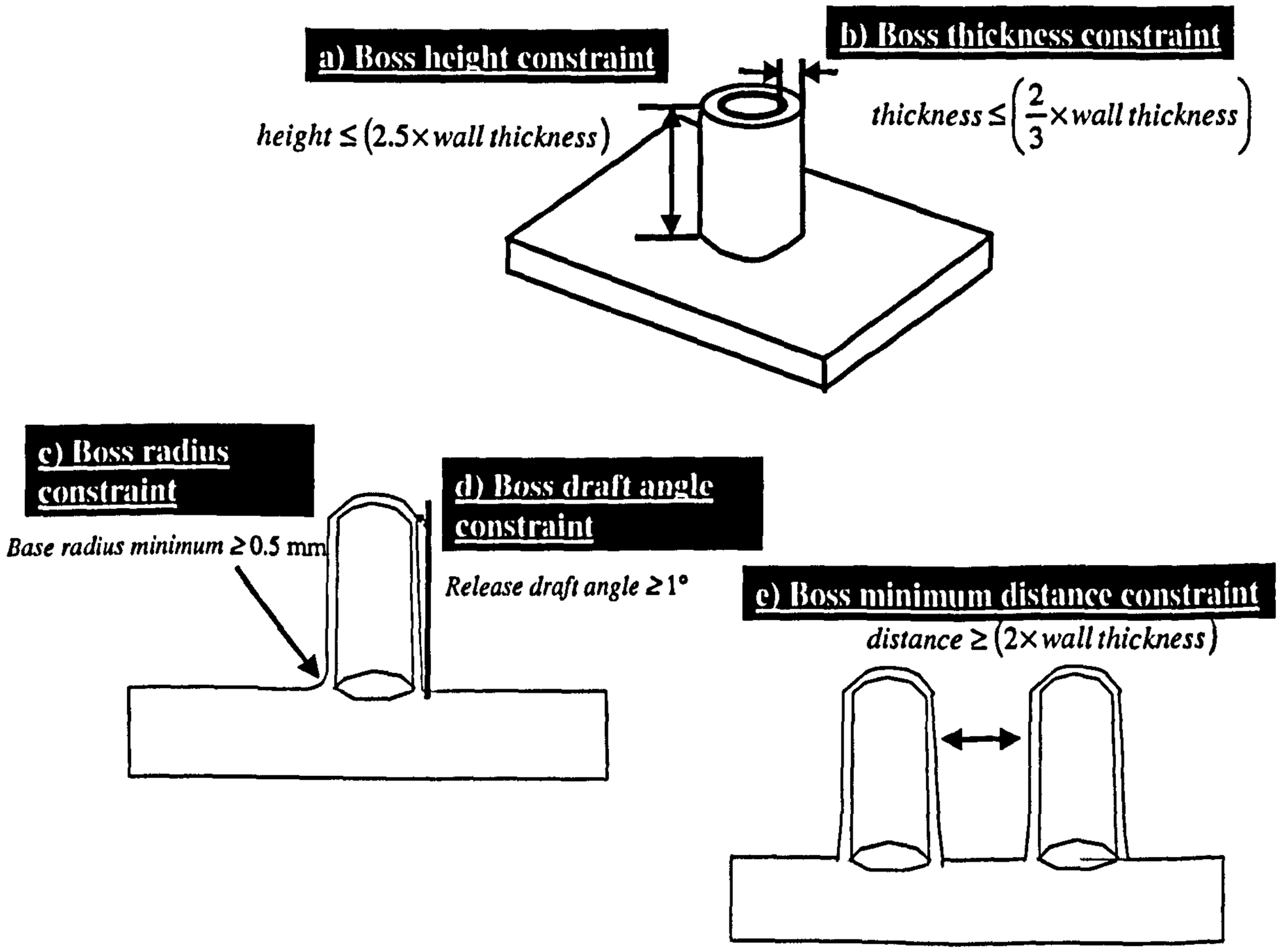


Figure 7.14 Capturing boss manufacturability constraints

7.4.1.2.3 Web constraints

Webs are reinforcing features used with both walls and bosses to further support and improve their structural integrity (Dow 2001). Webs are considered to be similar to rib reinforcements. It is for this reason, that some of the rib design constraints apply for the webs too. Figure 7.15 illustrates the web constraints that need to be considered.

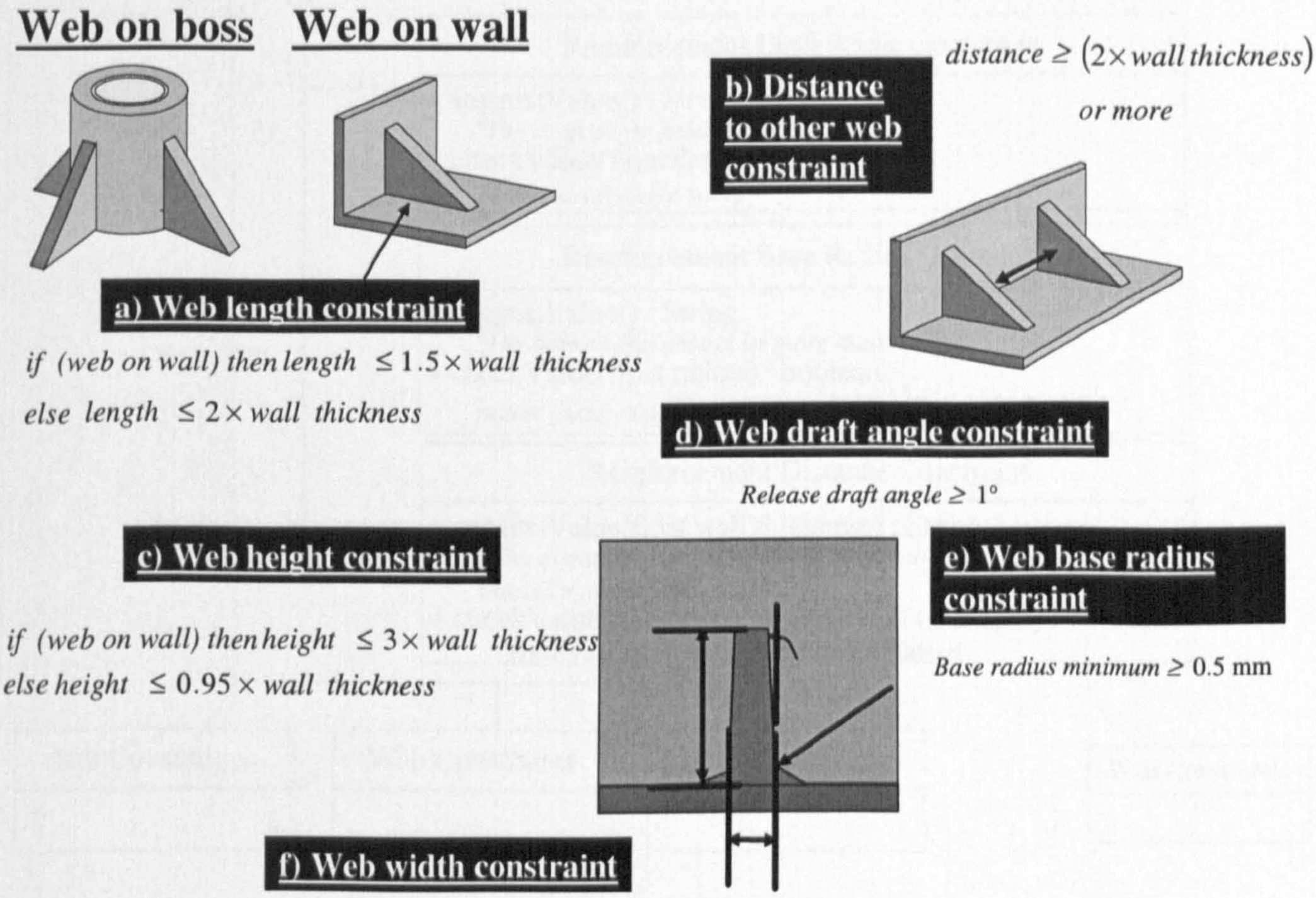


Figure 7.15 Capturing web manufacturability constraints

The constraints that the rib, web and boss have in common were represented as aggregated classes of the "Reinforcement Constraints" class. As figure 7.16 illustrates, the "Rib Constraints", "Boss Constraints" and "Web Constraints" classes inherit from the "Reinforcement Constraint" class. Furthermore, the latter has an aggregation relationship with the manufacturing constraints that are common to the three inherited classes. These are: "Reinforcement Draft Angle Constraint", "Reinforcement Base Radius Constraint"

and "Reinforcement Distance Constraint". Moreover, the "Reinforcement Distance Constraint" class has an interaction with the "Wall Constraints" class, as the distance to other reinforcement is dependent on the wall thickness. Therefore, the manufacturability of the wall should be firstly ensured.

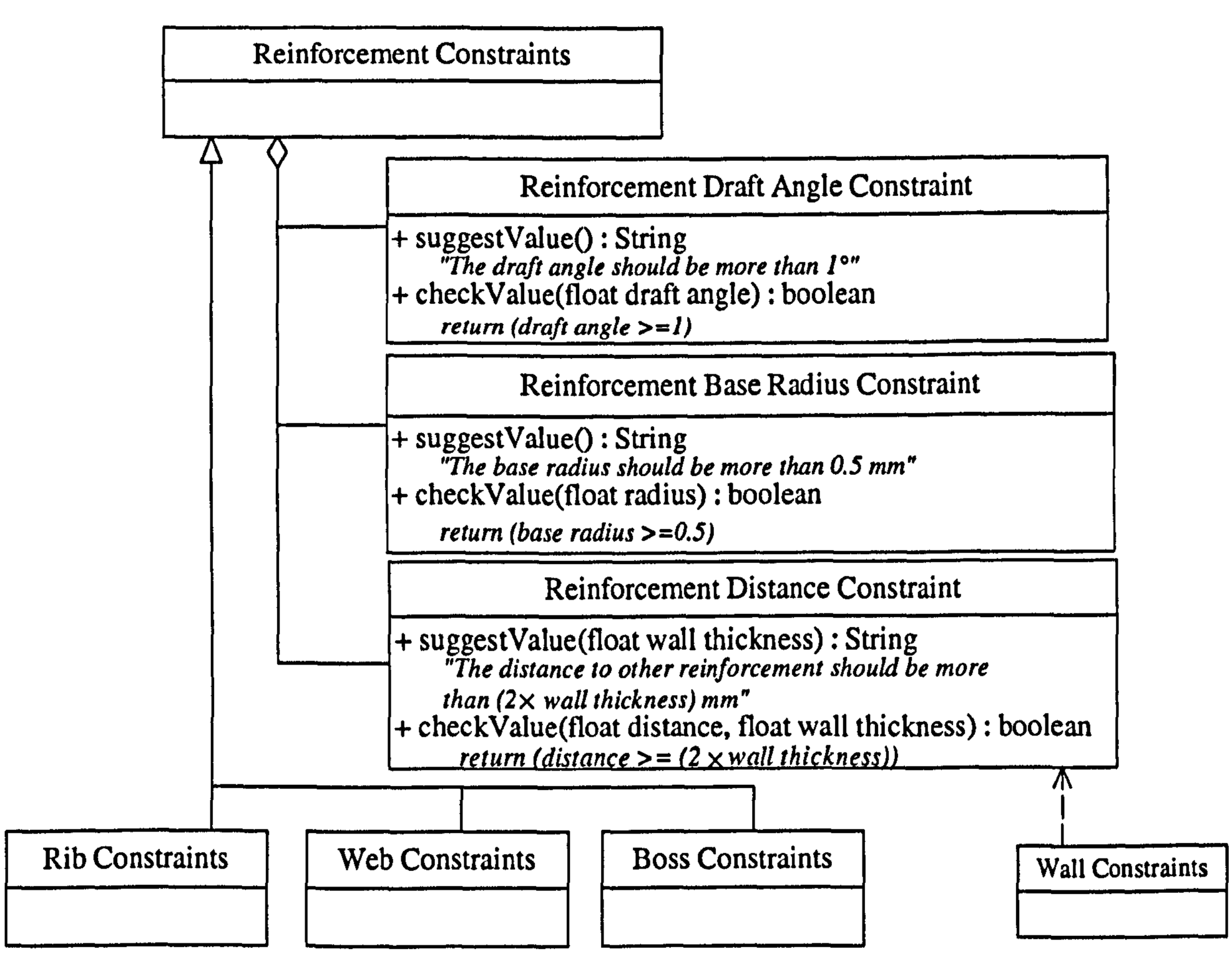


Figure 7.16 "Reinforcement Constraints" class representation using UML notation

7.4.1.3 Modelling of the hole constraints

Holes are a common feature in injection moulded parts. Several process constraints must be taken into account when designing holes (Al-Ashaab 1994):

- If it is a blind hole, its maximum depth must be less or equal to two times the diameter of the hole (see figure 7.17-a).
- As shown in figure 7.17-b, the distance between two holes or between one hole and the edge of the surface must be at least the diameter of the hole. Otherwise venting in

the mould is essential to avoid short shot problems and to ease the flow of the material.

- A draft angle of at least 1 degree is needed to ease the release from the mould in ejection if the hole is parallel to the injection moulding machine's axis (see figure 7.17-c).
- A hole should be avoided if possible if its direction is perpendicular to the injection machine's axis, as shown in figure 7.17-d. Otherwise, the cost of the mould will increase as special arrangements are required to eject the part from the mould such as using side core or multi plate mould.
- If a hole is placed on the centre of a rotational part, only the manufacturability of the wall on which it is placed need to be ensured.

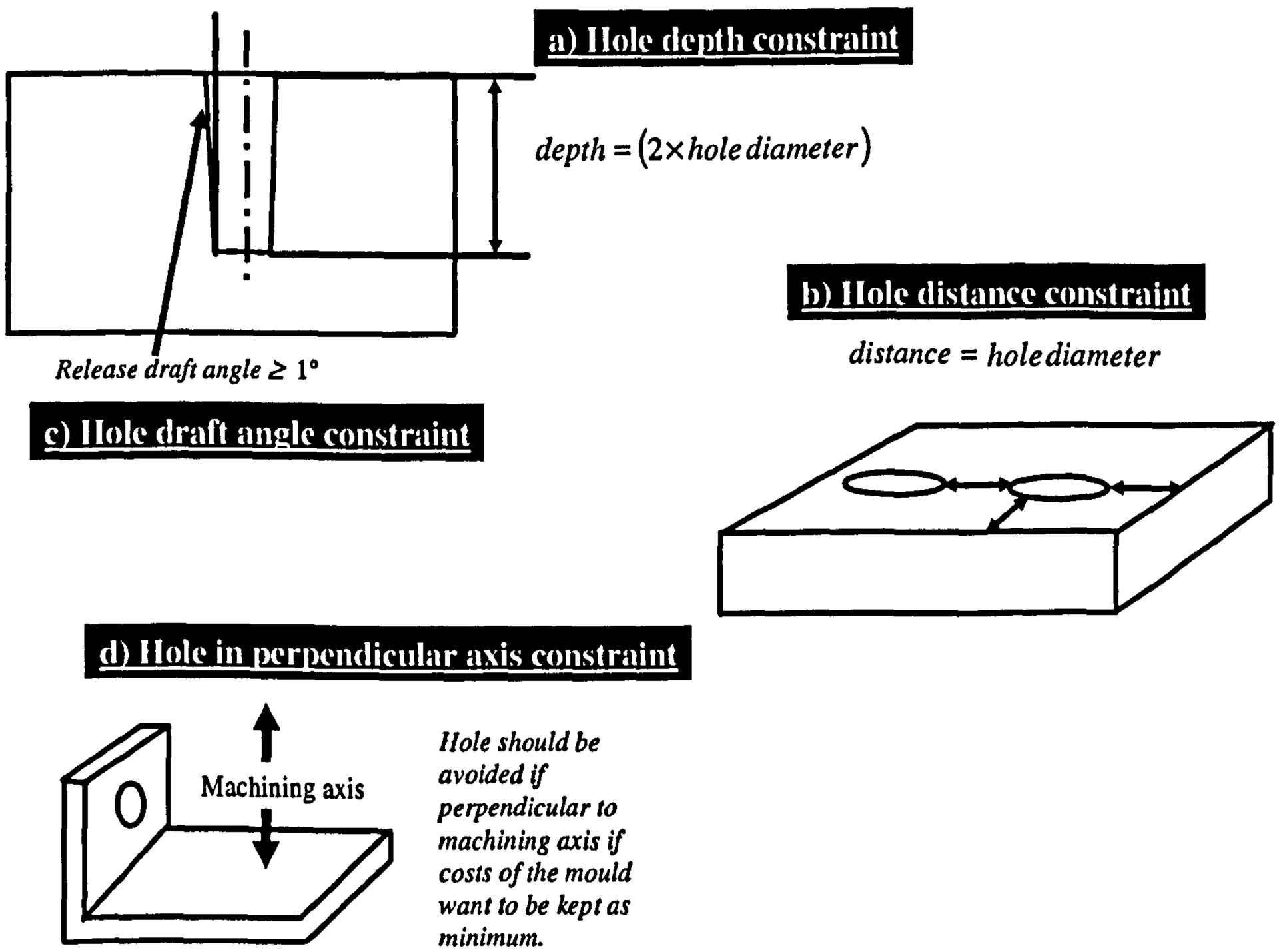


Figure 7.17 Capturing hole manufacturability constraints

7.4.1.4 Modelling of the corner shape design constraints

In the design of injection moulded parts, sharp corners must be avoided. Sharp corners, particularly inside corners, cause poor flow patters, reduced mechanical properties as well as stress concentration (Ticona 2000). To avoid these problems corners must be rounded by a suitable radius. This radius must be in the range of 25% to 75% of wall thickness; 50% is suggested (Dow 2001). The constraints for the different types of corners shapes that can be used are shown in figure 7.18-a,b,c,d.

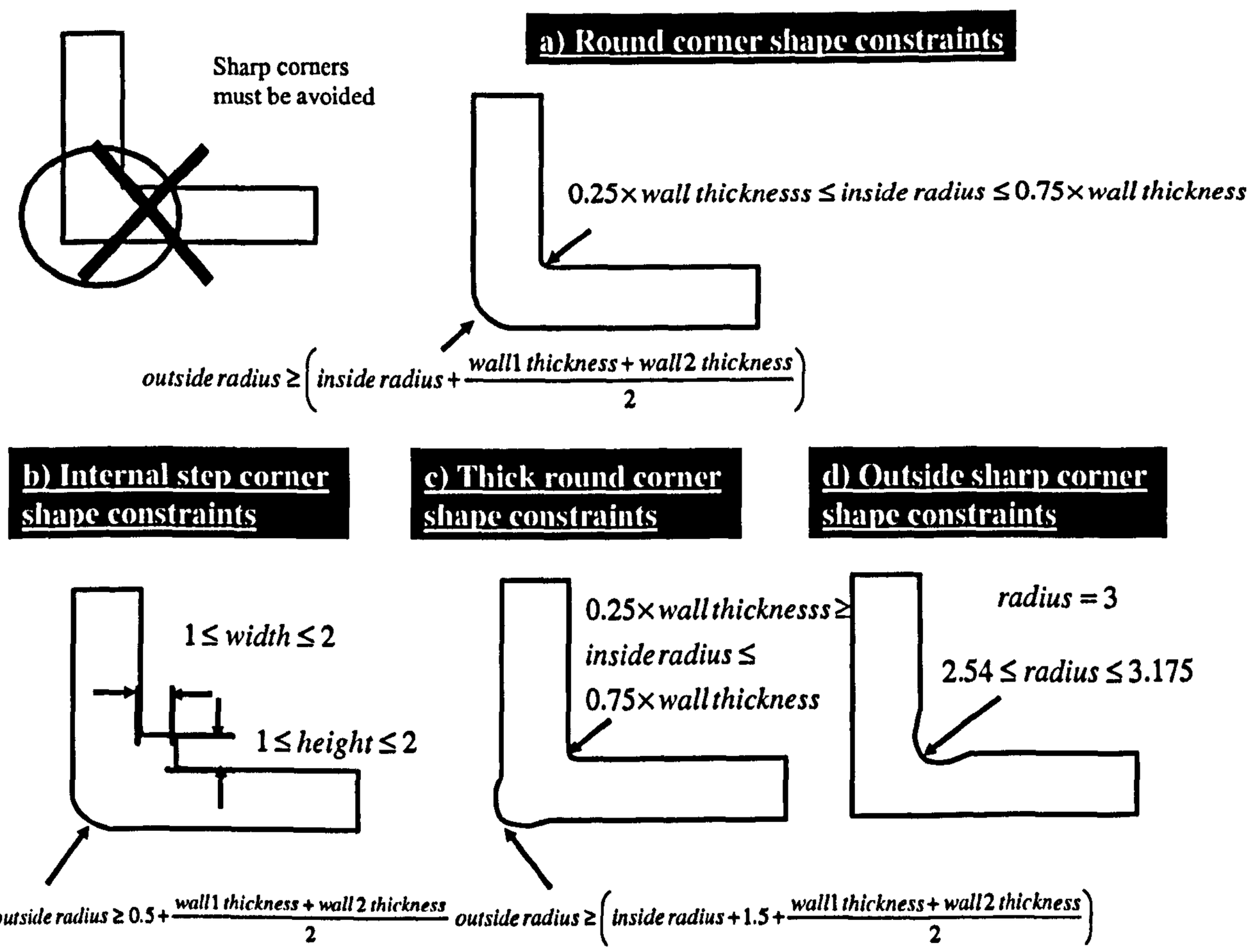


Figure 7.18 Capturing corner shape manufacturability constraints

7.4.1.5 Modelling of the text feature design constraints

During the design of a part, it is sometimes required to attach letters to a visible part surface. For example, the identification of the plastic material used. For this, the standard identification symbol of the Society for the Plastic Industry is written on a visible surface

(Dieter 2000). These printed letters must follow the following constraints (GE Plastics 1999):

- The raised or recessed letters placed on walls should have a minimum radius of 0.254 mm (see figure 7.19). This is because the breaking of the sharp edge helps the appearance of the lettering in the part.

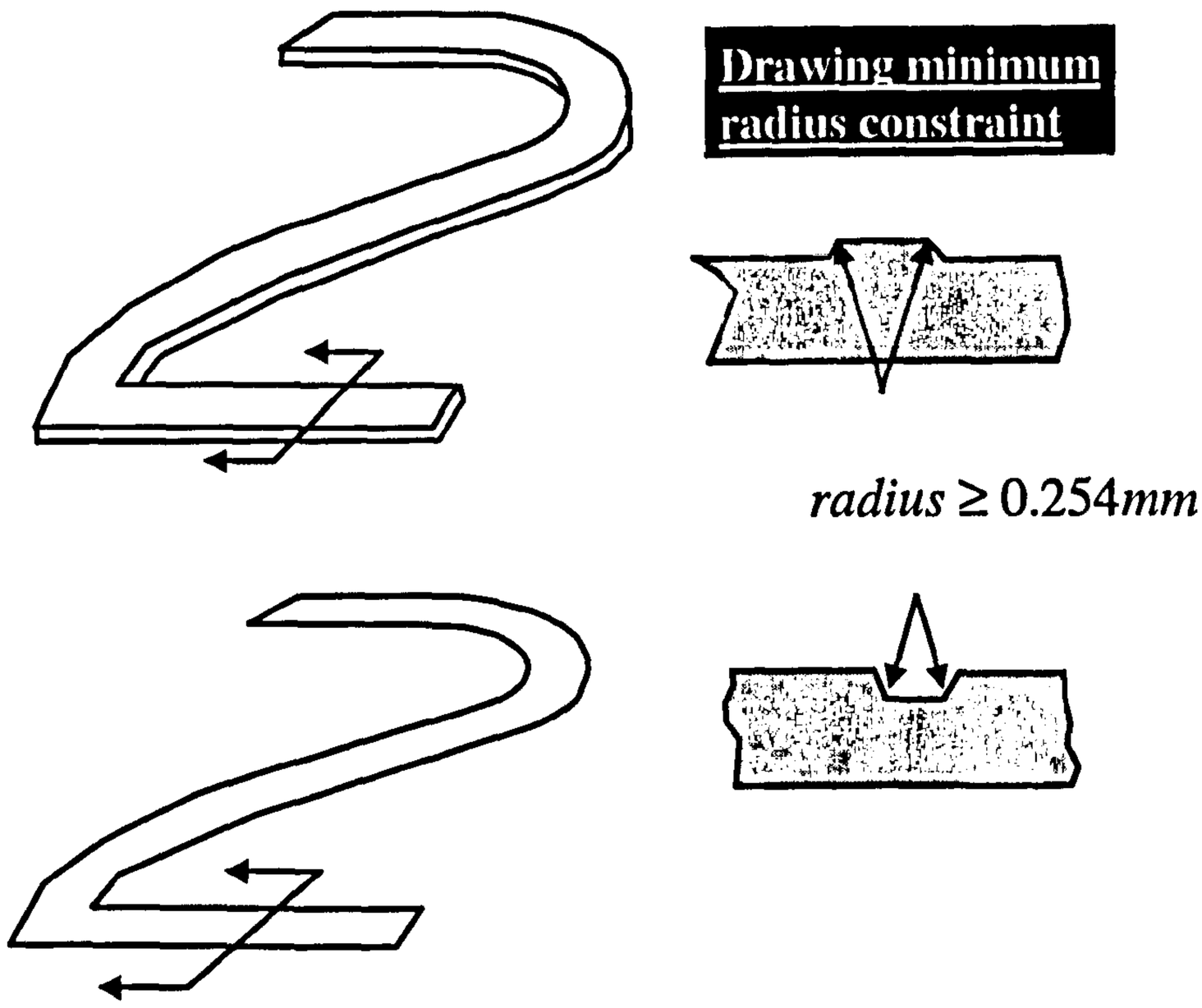


Figure 7.19 Capturing text feature manufacturability constraints

7.4.1.6 Modelling of the parting line, gating and ejection position constraints

7.4.1.6.1 Parting line constraints

A parting line is caused when the core and the cavity parts of the mould meet, as illustrated in figure 7.20-a. This leaves a line on the surface of the part that might affect its appearance if this aspect has not been considered during the design of the part. Therefore, the product engineer should collaborate with the toolmaker to determine its location.

7.4.1.6.2 Gating and ejection position constraints

The product engineer should also be aware that the location of gates and ejection pins has a critical effect on the part appearance (see figure 7.20-b,c). Their location is usually determined by the toolmaker during the design of the mould. However, it is critical that the product engineer and toolmaker have closer collaboration. The constraints related to positioning gates and ejections pins will be explained in more detail in section 7.4.4.5.2 and 7.4.4.3. However, a brief explanation is given below.

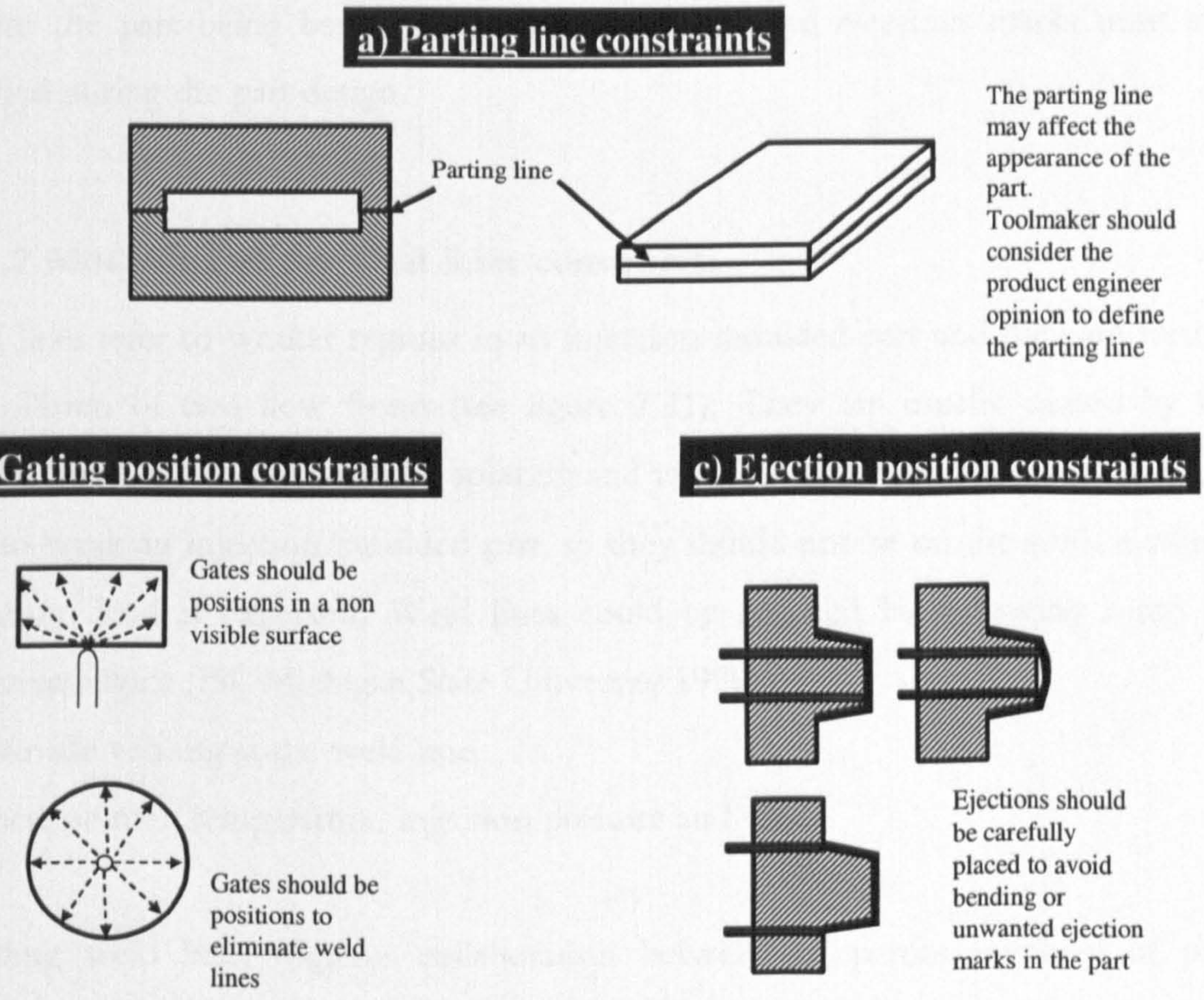


Figure 7.20 Capturing parting line, gating and ejection positions manufacturability constraints

When deciding the position of the gates in the part, several constraints must be taken into account (Dow 2001):

- Gates should be positioned on a non visible surface whenever possible.
- Gates should not be placed on a surface where a maximum load is expected.

- Gates should be placed in the thickest section of a part to ensure prevention of sinks and voids.
- When weld lines are unavoidable, the gate should be positioned at the farthest distance from an obstruction to minimise the weld line.
- In case of a rotational product, the best position is at the centre of the part to have an even flow of material.

In addition, the position of the ejections pins should be carefully consider in order to prevent the part being bent during ejection. Unwanted ejections marks must also be specified during the part design.

7.4.1.7 Modelling of the weld lines constraints

Weld lines refer to weaker regions in an injection moulded part and they are formed by the collision of two flow fronts (see figure 7.21). They are usually caused by having multiple gates in the mould or by splitting and rejoining flow fronts around inserts. They tend to weak an injection moulded part, so they should not be on the surface where the maximum load is expected. Weld lines could be reduced by following some of the following advice (ISL Michigan State University 1999):

- Provide venting at the weld line.
- Increase melt temperature, injection pressure and speed.

Avoiding weld lines requires collaboration between all parties involved in product development. This is because the product engineer knows where the maximum load is expected in the product. Moreover, the toolmaker and process engineer could take actions in order to avoid the weld lines.

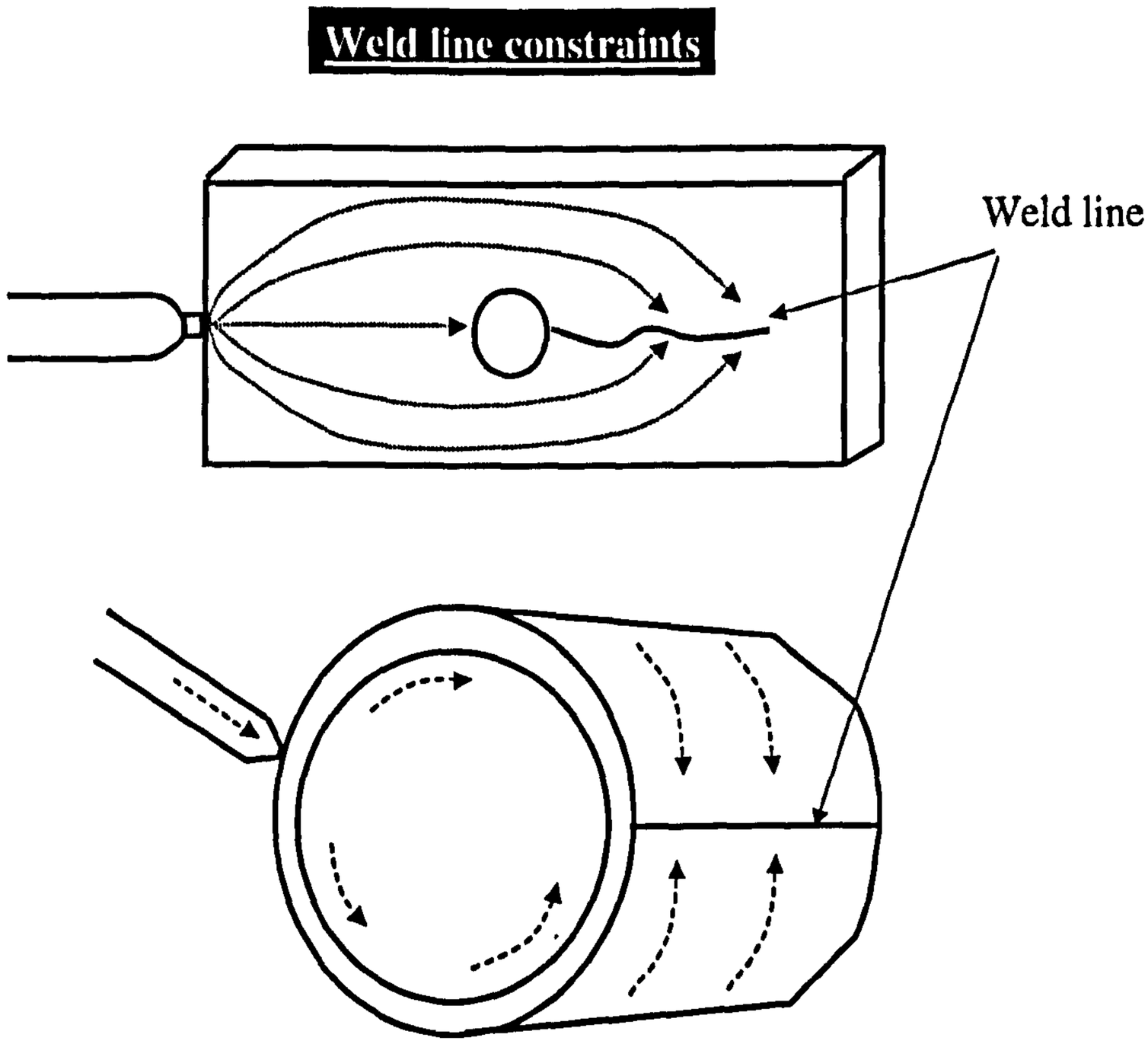


Figure 7.21 Capturing weld line manufacturability constraints

7.4.1.8 Modelling of the moulded in assembly system design constraints

Moulded in assembly systems are generally economical methods of assembly, since no additional fastener, adhesive or special equipment is required (Ticona 2000). The following moulded in assembly systems are considered:

7.4.1.8.1 Snap fit constraints

For high volume production, moulded in snap fits provide economic and rapid assembly. In all snap fit designs, some proportion of the moulded part must flex like a spring, usually past a designed in interference, and quickly return, or nearly return, to its original position to create an assembly between two or more parts (Ticona 2000). The key to successful snap fit design is to have sufficient holding power without exceeding the elastic or fatigue limits of the material. The considerations for the design of the cantilever beams,

which is a more common type of snap fit, are the following (Ticona 2000; Ticona and Tim Spahr 2000):

- In designing the snap beam, it is important to avoid sharp corners or structural discontinuities, as stress will concentrate in such areas. To avoid such problems, inside corners should be designed with a minimum radius of 0.508 mm. It is considered as a good design standard an inside radius of 50% the wall thickness (see figure 7.22-a).

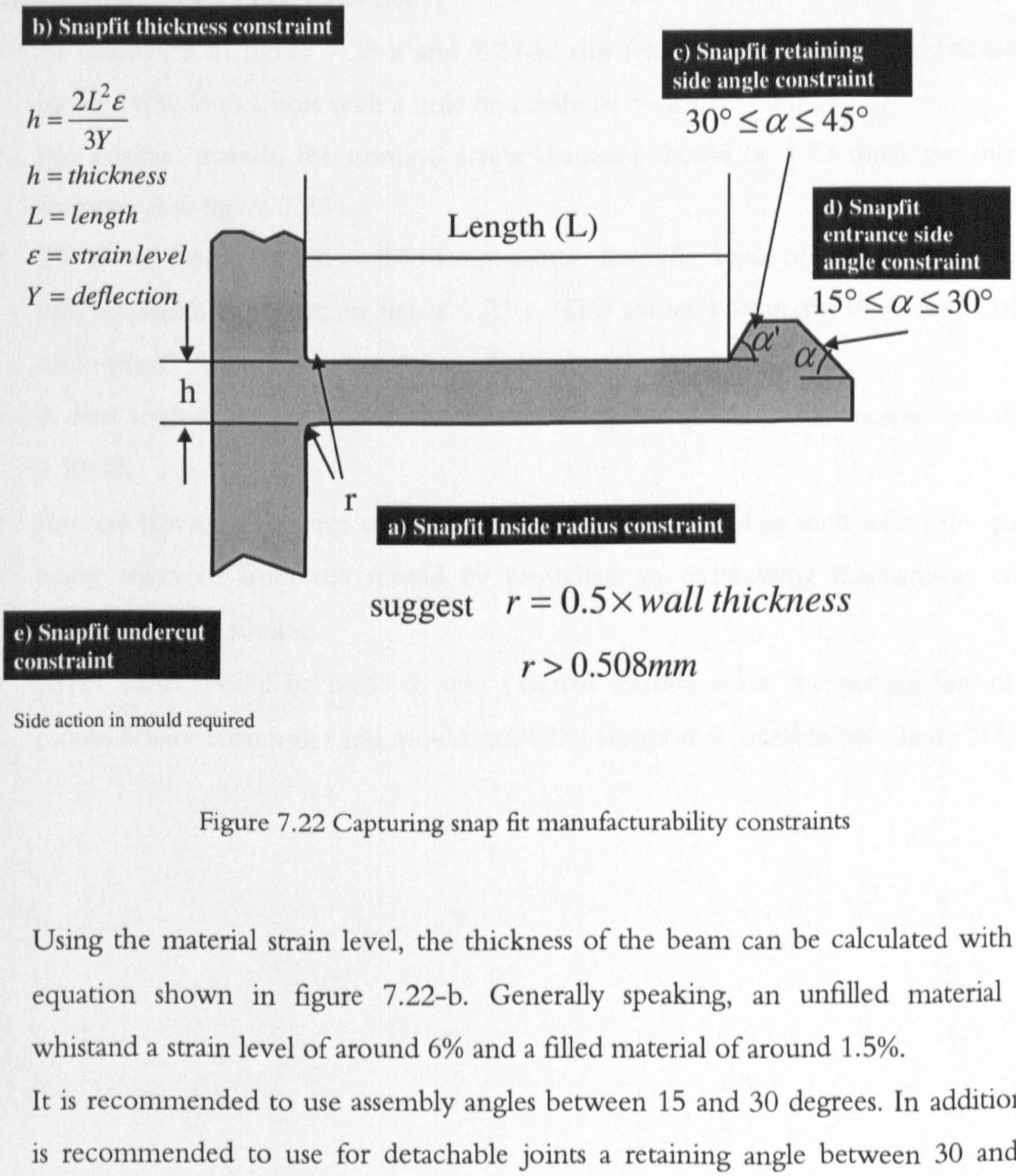


Figure 7.22 Capturing snap fit manufacturability constraints

- Using the material strain level, the thickness of the beam can be calculated with the equation shown in figure 7.22-b. Generally speaking, an unfilled material can whistand a strain level of around 6% and a filled material of around 1.5%.
- It is recommended to use assembly angles between 15 and 30 degrees. In addition, it is recommended to use for detachable joints a retaining angle between 30 and 45 degrees (see figure 7.22-c,d).

- Since the snap fit generally requires an undercut, a mould with side action is frequently required (see figure 7.22-e).

7.4.1.8.2 Moulded in thread constraints

In this type of assembly, mating male and female threads are moulded into the parts that are to be assembled. Several constraints should be taken into account when designing this feature (GE Plastics 1999; Dow 2001):

- As illustrated in figure 7.23-a and 7.23-b, this feature can be added as external or internal thread to a boss with a hole or a cylinder wall.
- For internal threads, the nominal screw diameter should be 1.25 times the internal diameter (see figure 7.23-c).
- The screw length should be 0.813 mm shorter than the depth of the cored hole when fully engaged, as shown in figure 7.23-e. This avoids bottoming the screw causing undo stress.
- A draft angle at the top of the boss or wall is a good lead in for the fastener (see figure 7.23-d).
- Internal threads also form undercuts and should be treated as such when the part is being removed from the mould by providing an unscrewing mechanism, which complicates the tooling.
- Every effort should be made to split external threads across the parting line of the mould where economics and mould reliability are most favourable (see figure 7.23-g).

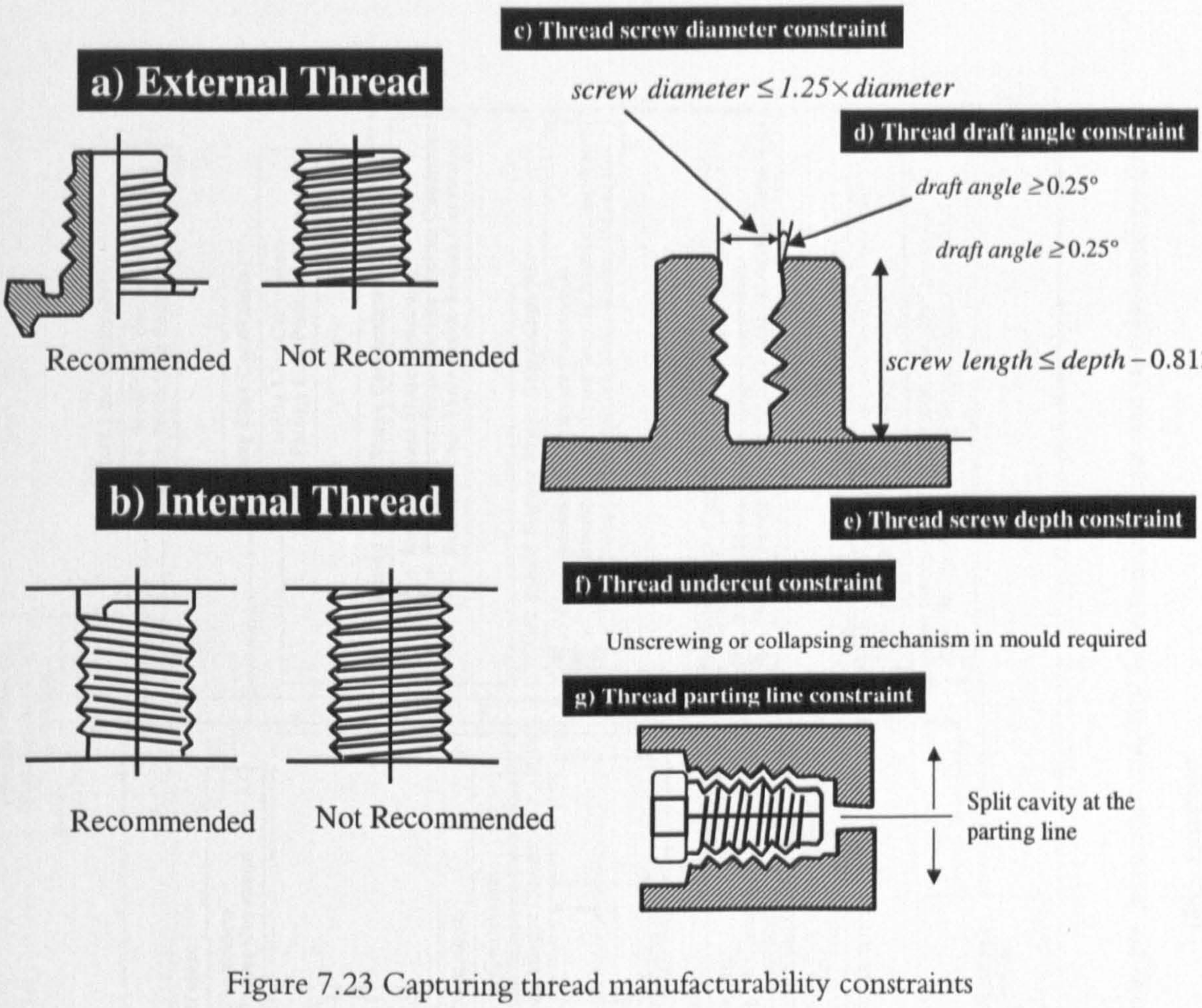


Figure 7.23 Capturing thread manufacturability constraints

7.4.1.9 Design features manufacturability constraints formal representation

After the manufacturing constraints for the different mouldable features were represented in classes, these were linked through aggregation relationships with a class called "Design Features Manufacturability Constraints". This class brings together the constraints that should be considered when designing a plastic part. Figure 7.24 illustrates this class and its relationships with the manufacturing constraints, which are represented using a package notation in order to group the interrelated classes.

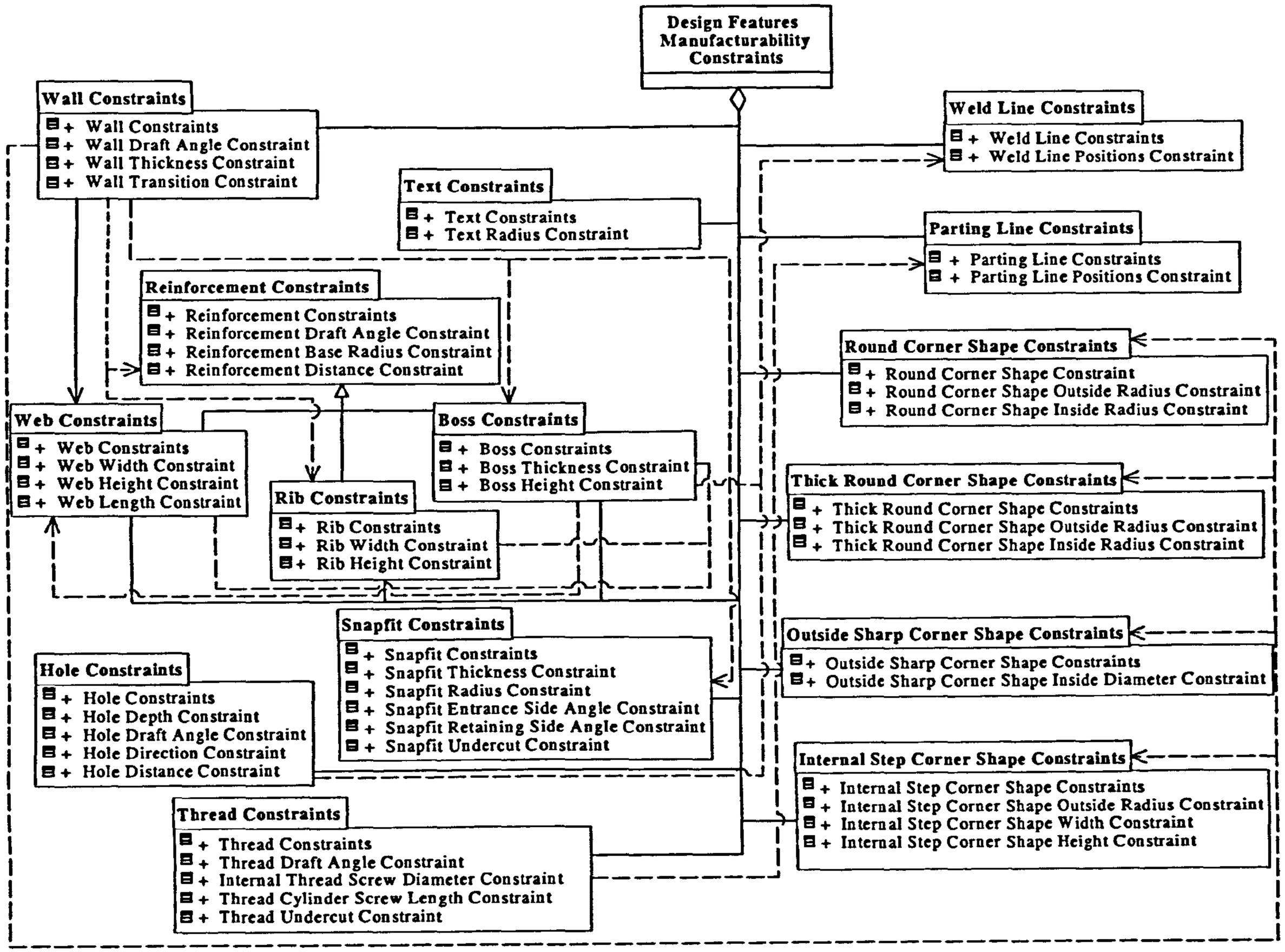


Figure 7.24 Representation of the "Design Features Manufacturability Constraints" class and its relationships using UML notation

Furthermore, the interactions among manufacturing constraints were represented using a dotted line connected among the different packages. These interactions represent the knowledge integrity of the model.

7.4.2 Modelling of the production equipment selection constraints

The optimum selection of an injection moulding machine for the production of an injection moulded part has a major impact on product quality and cost. The machine characteristics considered for this selection are: injection pressure, machine size, minimum mould thickness, maximum open distance, shot size and tie bar space (Al-Ashaab 1994; Menges, Michaeli et al. 2001). The identification, capture and representation of the production equipment selection constraints is explained in more detail in the following subsections.

7.4.2.1 Injection machine size constraint

The machine size is the maximum force the machine is capable of to keep the mould closed against the cavity pressure during injection. Insufficient clamping force gives rise to flash at the mould joint (Dow 2001). The sufficient clamping force is proportional to the projected area of the cavity. Therefore, the suitability of the size of a machine can be validated with the following formula (Al-Ashaab 1994):

Machine size $\geq$ *projection area* x *number of cavities* x *plastic tonnage required*

The same formula can be used for suggesting a possible machine size:

Machine size required = *projection area* x *number of cavities* x *plastic tonnage required*

In order to formally represent these formulas, they were modelled using UML object oriented language. As such, the constraint was represented in a class called "Machine Size Constraint", as shown in figure 7.25. The formulas were represented in the methods of the class as follows:

- The method "suggestValue" represents the rule to suggest the suitable machine size.

- The method "checkValue" represents the rule to ensure that the actual size of a machine is suitable for a particular product

Furthermore, the size of a machine should be considered after the manufacturability of the plastic part features has been ensured. Otherwise, it is possible that the machine size is being calculated with data that is outside manufacturing constraints. This will generate problems with the part during production and will also compel the recalculation of the machine size once the part data is modified. In order to prevent this from happening, the knowledge related to the part manufacturability, which belongs to the product engineering company, should also be considered by the process engineer. This interaction is represented using a dotted arrow from the "Design Features Manufacturability Constraints" to the "Machine Size Constraint".

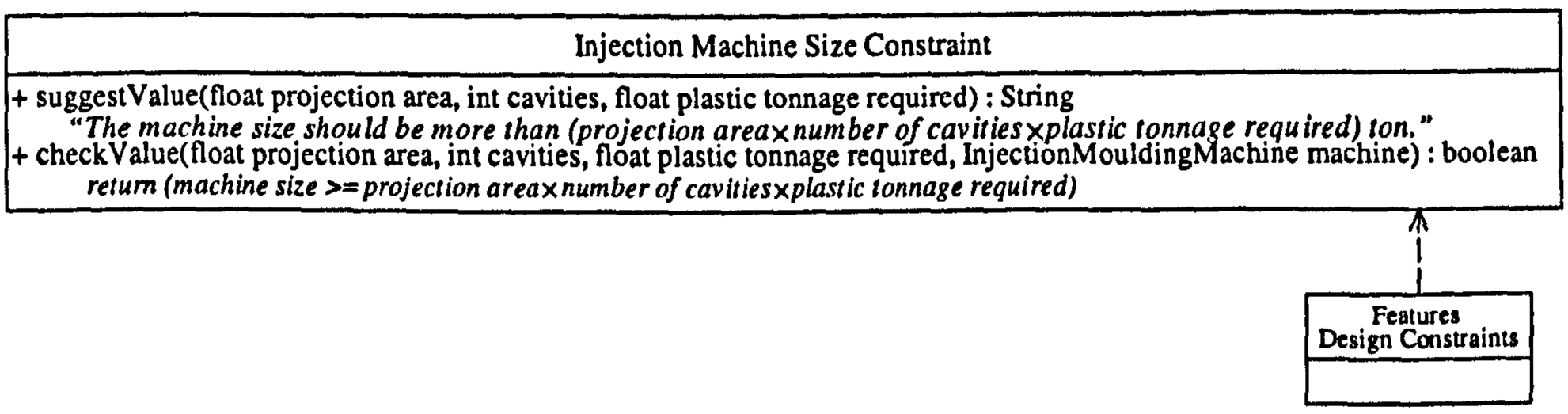


Figure 7.25 "Injection Machine Size Constraint" class representation using UML notation

7.4.2.2 Machine injection pressure constraint

The injection pressure that is required in a machine depends on the recommendation of the material supplier. As such, the following formulas were captured:

- To ensure an adequate injection pressure:

$$\textit{Machine injection pressure} \geq \textit{Plastic injection pressure}$$

- To suggest a suitable injection pressure:

$$\textit{Machine injection pressure required} = \textit{Plastic injection pressure}$$

Figure 7.26 illustrates the "Injection Machine Injection Pressure Constraint" class, which represents these formulas in the methods "suggestValue" and "checkValue".

Injection Machine Injection Pressure Constraint
+ suggestValue(float plastic injection pressure) : String *"The injection pressure should be more than (plastic injection pressure) Pa."* + checkValue(float plastic injection pressure, InjectionMouldingMachine machine) : boolean *return (machine injection pressure >= plastic injection pressure)*

Figure 7.26 "Injection Machine Injection Pressure Constraint" class representation using UML notation

7.4.2.3 Injection machine shot size constraint

The machine's shot size must be enough to fill all the mould cavities in order to avoid short shot problems. This can be determined with the formula:

$$\textit{Machine shot size} \geq \textit{part volume} \times \textit{number of cavities}$$

This formula can also be used to suggest a possible optimal machine shot size:

$$\textit{Machine shot size required} = \textit{part volume} \times \textit{number of cavities}$$

These formulas are formally represented in the methods of the "Injection Machine Shot Size Constraint", as figure 7.27 shows. An interaction with the "Design Features Manufacturability Constraints" class is also represented, as the features need to be within manufacturing constraints in order to be considered for the volume calculation.

Injection Machine Shot Size Constraint
+ suggestValue(float part volume, int cavities) : String *"The shot size should be more than (part volume×number of cavities) mm3"* + checkValue(float part volume, int cavities, InjectionMouldingMachine machine) : boolean *return (machine shot size >= (part volume ×number of cavities))*

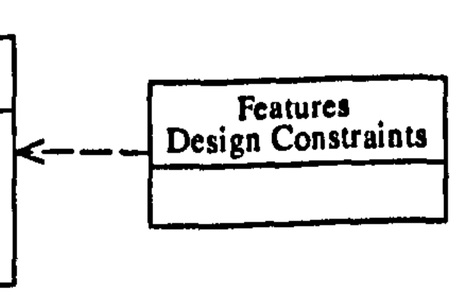


Figure 7.27 "Injection Machine Shot Size Constraint" class representation using UML notation

7.4.2.4 Mould thickness constraint

The thickness of the mould must be between two machine's characteristics: minimum mould thickness and maximum open distance. The formulas are represented as follows:

- To ensure the machine is suitable for a specific mould:

$$Minimum\ mould\ thickness \leq mould\ thickness \leq Maximum\ open\ distance$$

- To suggest a minimum mould thickness:

$$Minimum\ mould\ thickness = Mould\ thickness$$

In case that the mould has not been defined, this formula is not taken into account. This is because the thickness of the mould cannot be known before the mould has been designed by the toolmaker. The "Machine Mould Thickness Constraint" class, which represents this consideration, is illustrated in figure 7.28.

Injection Machine Mould Thickness Constraint
+ suggestValue(float mould thickness) : String *"The machine minimum mould thickness should be less and the maximum open distance should be more than (mould thickness) mm."* + checkValue(float mould thickness, InjectionMouldingMachine machine) : boolean *return (machine minimum mould thickness <= (mould thickness) <= machine maximum open distance)*

Figure 7.28 "Injection Machine Mould Thickness Constraint" class representation using UML notation

7.4.2.5 Injection machine tie bar space constraint

The space between the tie bars of a machine must be enough to fit the mould. The captured formulas are:

$$Injection\ moulding\ machine\ horizontal\ tie\ bar\ space \geq (injection\ mould\ width + 10)$$

and

$$Injection\ moulding\ machine\ vertical\ tie\ bar\ space \geq (injection\ mould\ length + 10)$$

These formulas can also be used to suggest the required horizontal and vertical tie bar space:

$$Injection\ moulding\ machine\ horizontal\ tie\ bar\ space = (injection\ mould\ width + 10)$$

and

$$Injection\ moulding\ machine\ vertical\ tie\ bar\ space = (injection\ mould\ length + 10)$$

Figure 7.29 illustrates the classes that represent the previous formulas. These formulas depend on the length and width of a mould. If the mould has not been defined as yet, it is possible to calculate an approximate length and width using the quantity and layout of the mould's cavities. The knowledge required to perform these calculations belongs to the toolmaker (see section 7.4.4.1.1) and would be shared with the process engineer for the calculation of the required tie bar space. This is represented with an interaction between the "Injection Machine Vertical Tie Bar Space Constraint" and the "Injection Mould Plate Length Constraint" class as well between the "Injection Machine Horizontal Tie Bar Space Constraint" and the "Injection Mould Plate Width Constraint" class. The collaboration of both toolmaker and process engineer is further required in order to agree on a suitable number and layout of cavities.

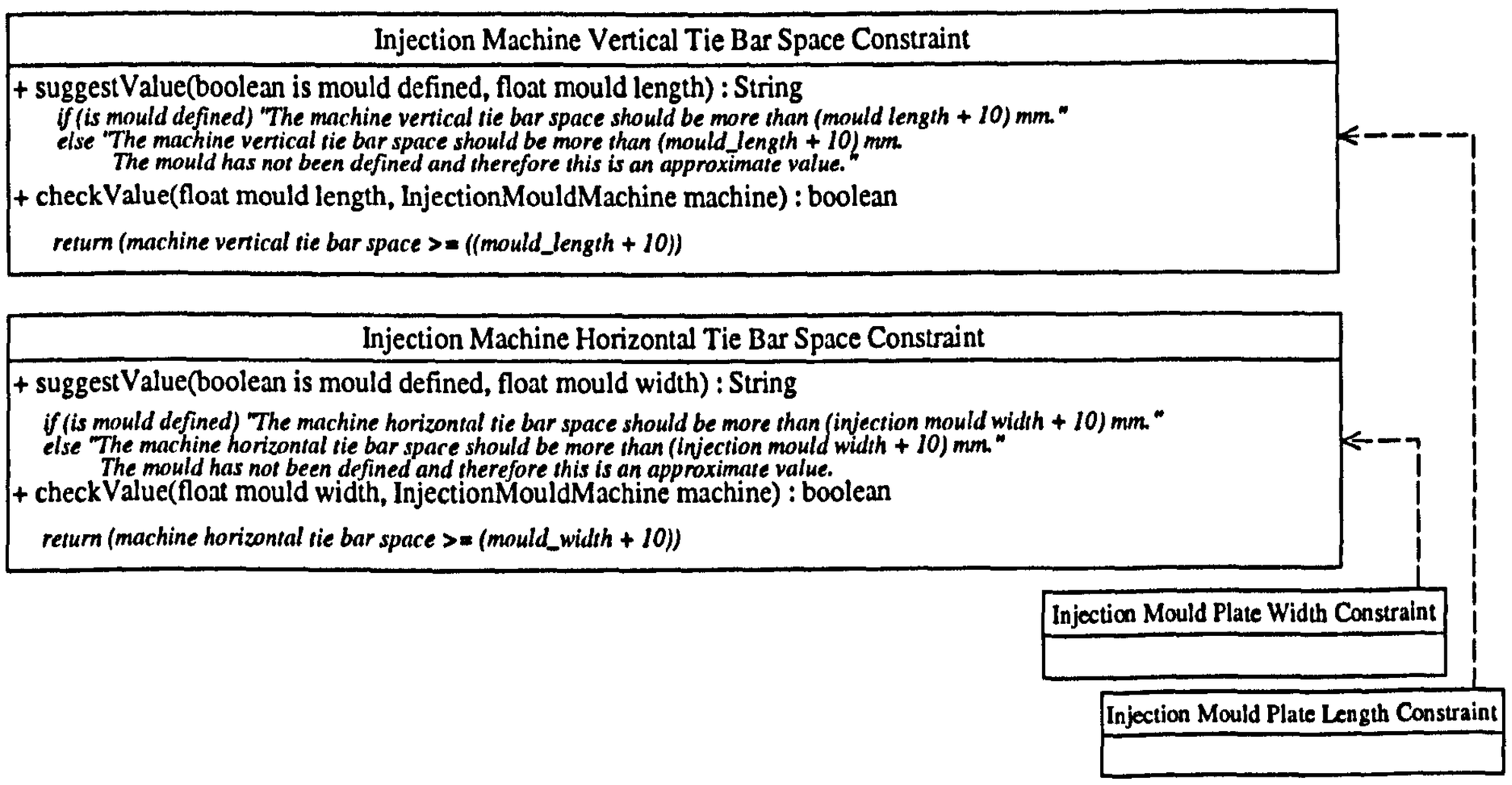


Figure 7.29 "Injection Machine Vertical Tie Bar Space Constraint" and "Injection Machine Horizontal Tie Bar Space Constraint" class representation using UML notation

After representing the considerations for all the machine characteristics, these were brought together through aggregation relationships with the "Production Equipment Selection Constraints" class. Figure 7.30 illustrates this class, which represents the knowledge required to ensure a machine is suitable to produce a part or to suggest a suitable machine. Furthermore, the interactions among manufacturing constraints were

represented using dotted arrows. For example, the "Injection Machine Vertical Tie Bar Space Constraint" has an interaction with the toolmaker's knowledge related to the mould plate length calculation.

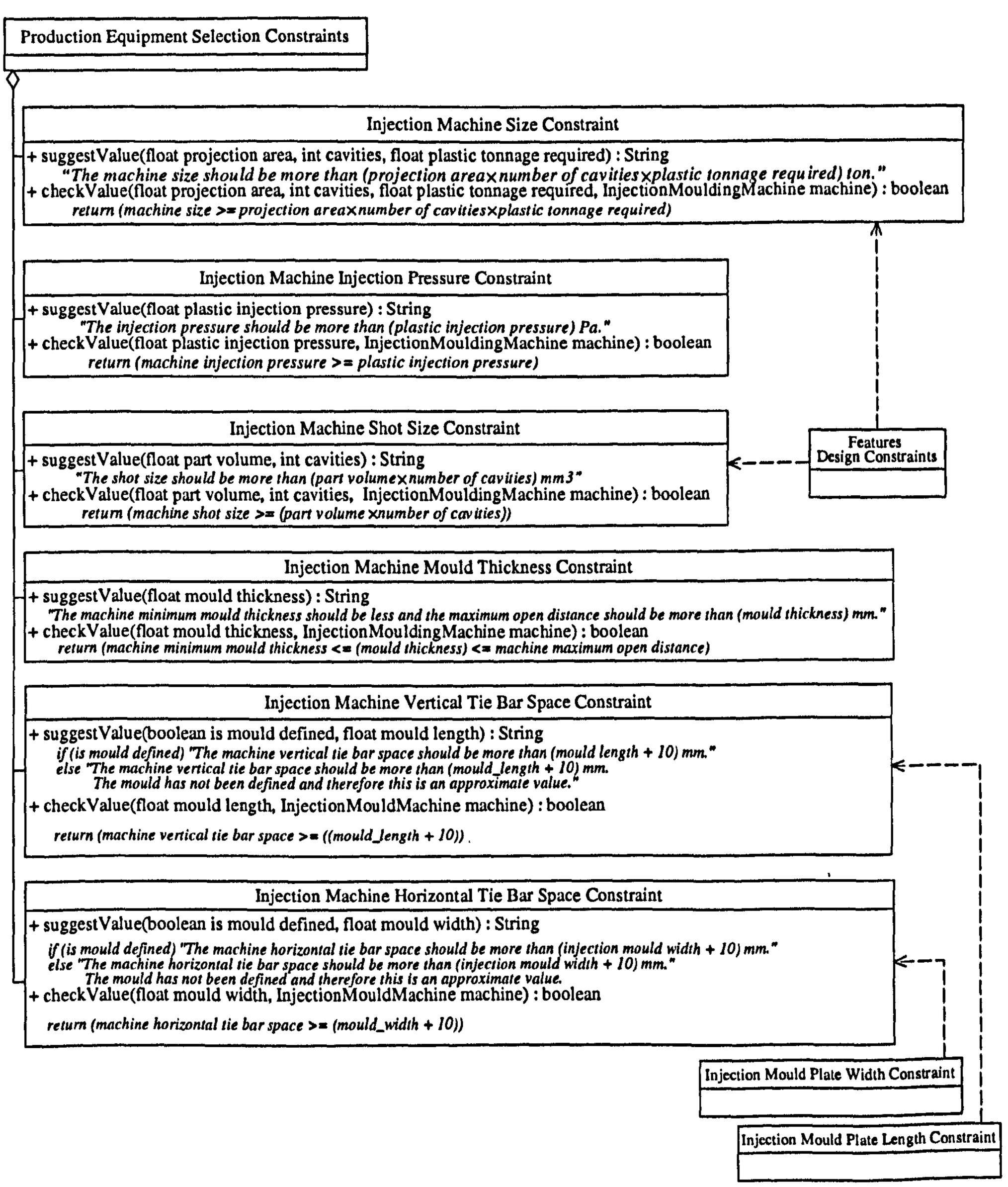


Figure 7.30 Representation of the "Production Equipment Selection Constraints" class and its relationships using UML notation

In order to enable the selection of a suitable machine, a list of the company's resources should be made available. For this reason, an Engineering Data Model with the injection machines' capabilities was produced. This model is referred to as Resources Model. As figure 7.31 shows, the model contains a class called "Injection Moulding Machine", which inherits from the "Resource" class. The "Injection Moulding Machine" class contains different attributes, which describe the capacities of a machine, such as name, description, machine size, shot size as well as horizontal and vertical tie bar space.

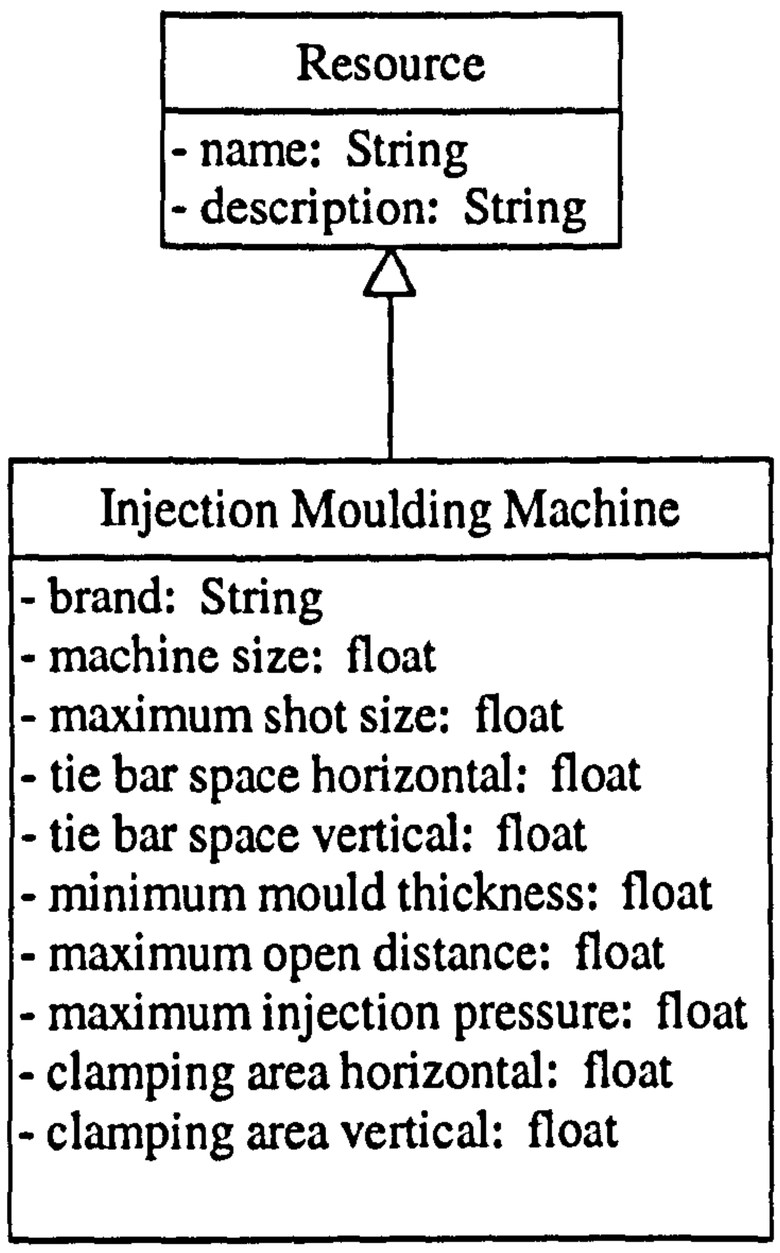


Figure 7.31 Resources Model representation using UML notation

7.4.3 Modelling of the process parameters selection constraints

There are several processing variables that need to be controlled in the injection machine in order to improve the part quality, reduce part variations, and increase overall productivity. These variables are (Miller 1996; Osswald, Turng et al. 2002):

- Injection velocity: is the rate at which the screw moves. This is the most critical variable during fill. A polymer flows more easily as injection velocity is increased. However, injection velocity that is too high can create excessive shear and result in

problems such as splay and jetting. The injection velocity is dependent of the type of material being moulded. High velocity is desired if a homogenous filling is required. Low injection velocity is often desired to achieve high quality surfaces and to avoid viscous dissipation and jetting during mould filling.

- Processing temperature: The material temperature is one of the main parameters that need to be set on the injection moulding machine. If the temperature setting is too high, the material can thermally degrade; if melt temperature is too low, the process can result in short shot. To achieve the best result the processing temperature should be maintained within the range recommended by the supplier.

- Injection pressure: Injection pressure is the primary variable of concern during the pack phase. The screw maintains pressure in the melt, compensating for shrinkage, which could cause sinks and voids. The injection pressure is dependent on the type of material being moulded.

- Pack hold pressure: After the mould is packed, the plastic is held in the mould until it is partially solidified and the gate freezes. The drop in injection pressure reflects the amount of shrinkage that occurs from cooling. This pressure is also dependent on the type of material being moulded.

- Cooling time: Cooling is generally the longest part of the moulding cycle, up to 80% of the cycle time. Because the gates are sealed during this phase, cooling temperature and time are the only variables at work. The following equation gives a rough estimate of the minimum cooling time needed before part ejection (Menges, Michaeli et al. 2001):

$$t_c = \frac{h^2}{\alpha\pi^2}\ln\left|\frac{4}{\pi}\left(\frac{T_M - T_w}{T_E - T_W}\right)\right|$$

α = *thermal diffusivity of the material*

h = *wall thickness*

T_w = *mould wall temperature*

T_M = *melt temperature*

T_E = *ejection temperature*

Process Parameters Selection Constraints

Injection Velocity Constraint

+ suggestValue(float plastic injection velocity, boolean holes, boolean bosses) : String
"The recommended injection velocity is (plastic injection velocity) m/s."
if (holes) or (bosses) "Increase injection velocity to avoid weld lines."

Plastic Processing Temperature Constraint

+ suggestValue(float plastic processing temperature min, float plastic processing temperature max, boolean holes boolean bosses) : String
"The processing temperature should be between (plastic processing temperature min) °C and (plastic processing temperature max) °C.
Recommended: $\left(\frac{\text{plastic processing temperature min} + \text{plastic processing temperature max}}{2}\right)$ *°C."*
if (holes) or (bosses) "The processing temperature could be increased to avoid weld lines"
+ checkValue(float processing temperature, float plastic processing temperature min, float plastic processing temperature max) : double
return (plastic processing temperature min <= processing temperature <= plastic processing temperature max)

Injection Pressure Constraint

+ suggestValue(float plastic inj pressure min, float plastic inj pressure max, boolean holes, boolean bosses) : String
"The injection pressure should be between (plastic injection pressure min) Pa and (plastic injection pressure max) Pa.
Recommended: $\left(\frac{\text{plastic injection pressure min} + \text{plastic injection pressure max}}{2}\right)$ *Pa."*
if (holes) or (bosses) "The injection pressure could be increased to avoid weld lines"
+ checkValue(float injection pressure, float plastic injection pressure min, float plastic injection pressure max) : boolean
return (plastic injection pressure min <= injection pressure <= plastic injection pressure max)

Pack-Hold Pressure Constraint

+ suggestValue(float plastic packhold pressure min, float plastic packhold pressure max) : String
"The pack-hold pressure should be between (plastic packhold pressure min) Pa and (plastic packhold pressure max) Pa.
Recommended: $\left(\frac{\text{plastic packhold pressure min} + \text{plastic packhold pressure max}}{2}\right)$ *Pa."*
+ checkValue(float packhold pressure, float plastic packhold pressure min, float plastic packhold pressure max) : boolean
return (plastic packhold pressure min <= packhold pressure <= plastic packhold pressure max)

Cooling Time Constraint

+ suggestValue(float first wall thickness, float plastic thermal diffusivity, float mould wall temperature min, float mould wall temperature max, float, plastic processing temperature, float ejection temperature min, float ejection temperature max) : String

The recommended cooling time is:

$$t_c = \frac{\text{first wall thickness}^2}{\text{plastic termal diffusivity} \times \pi^2}\ln\left|\frac{4}{\pi}\left(\frac{\text{average mould wall temperature} - \text{plastic processing temperature}}{\text{average ejection temperature} - \text{average mould wall temperature}}\right)\right|$$

$$\text{average mould wall temperature} = \left(\frac{\text{mould wall temperature min} + \text{mould wall temperature max}}{2}\right)$$

$$\text{average ejection temperature} = \left(\frac{\text{ejection temperature min} + \text{ejection temperature max}}{2}\right)$$

Figure 7.32 Representation of the "Process Parameters Selection Constraints" class and its relationships in UML notation

The previous considerations for selecting the suitable process parameters were represented in different UML classes, such as "Injection Velocity Constraint", "Plastic Processing Temperature Constraint", "Injection Pressure Constraint", "Pack Hold Pressure Constraint" and "Cooling Time Constraint" (see figure 7.32). These classes were brought together through aggregation relationships with the "Process Parameters Selection Constraints" class. Furthermore, the interaction between the "Cooling Time Constraint" class and the "Wall Constraints" class is used to represent that the consideration of the cooling time should be done after the manufacturability of the walls in the part has been ensured.

7.4.4 Modelling of the injection mould design constraints

The mould with two plates is the simplest of all the mould design configurations. This is constructed from two distinct half units: the core and the cavity half. The point at which the two halves interface is known as the split or parting line. The core half of the mould is usually attached to the moving platen on the injection machine because the ejection system is commonly positioned behind this plate (see figure 7.33). On the other side, the cavity half of the mould is attached to the fixed platen of the machine directly in the front of the machine injection unit for material feeding of the mould (Cracknell and Dyson 1993).

In order to represent the constraints imposed on the mould design, the mould was decomposed in its different components: cooling system, ejection system, venting system and feed system, which includes runner system, gating system and sprue. The following subsection presents the identification, capture and representation of the mould design constraints. Thereafter, the following sections present the identification of the design constraints of the mould components. Their capture and representation is presented in Appendix E.

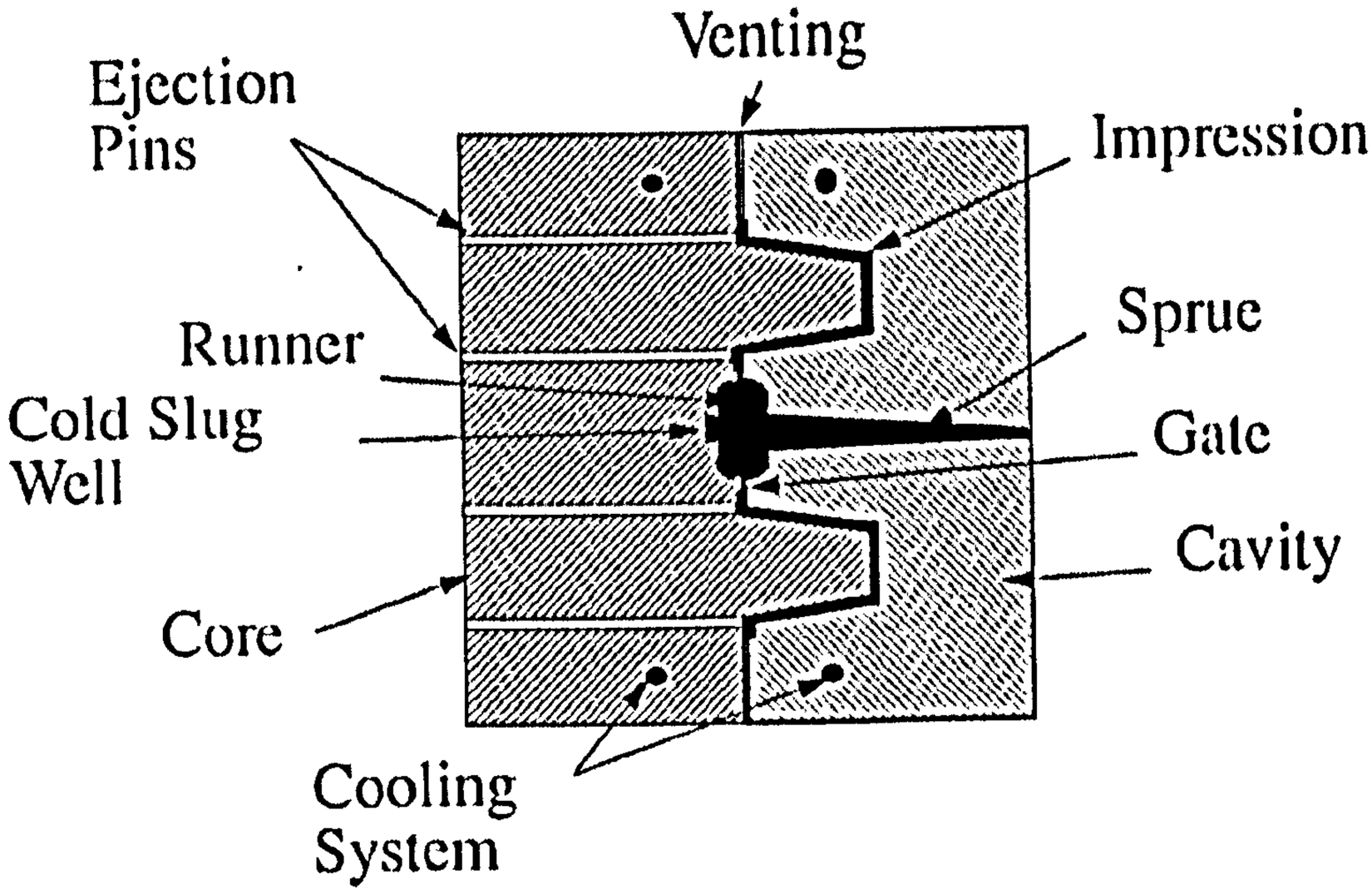
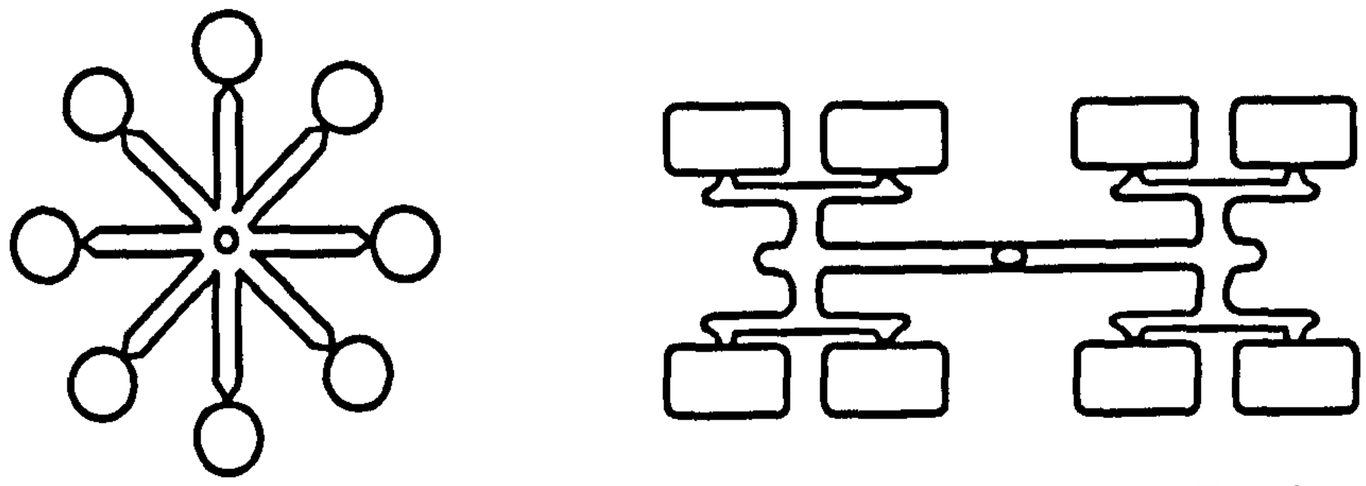


Figure 7.33 Components of the core and cavity in an injection mould

7.4.4.1 Modelling of the mould plates design constraints

The shape of the plastic part, which is referred to as cavity, is formed by two plates in the mould: the core and the cavity plate. The cavity plate shapes the outside form of the part and the core plate forms the inside shape. The methods used to incorporate these plates to the mould are known as integer or insert/bolster method. The integer method is when the cavity and core are machined from steel plates, which become part of the structural build up of the mould. The insert/bolster method is when the cavity and core are machined from small blocks of steel. These blocks are known as the core and cavity insert. They are then inserted and securely fitted into holes in a substantial block or plate of steel known as the bolster (Pye 1989).



a) Balanced Arrangement Type 1

b) Balanced Arrangement Type 2

c) Unbalanced Arrangement Type 1

d) Unbalanced Arrangement Type 2

Figure 7.34 Types of cavity arrangements in an injection mould plate

In the core and the cavity plates, the cavities are arranged in one of the following layouts:

- Balanced: This means that the distance the plastic material travels from the sprue to the gate is the same for each cavity. In this research, two types of balanced arrangement were considered. They are shown in figure 7.34-a,b.
- Unbalanced: This arrangement is used when a large number of cavities have to be accommodated, or where cavities are of greatly dissimilar shape. Figure 7.34-c and d illustrates the two types of unbalanced arrangements considered.

The design considerations for the mould plates are explained in detail in the following paragraphs. These are: injection mould plate length and width constraint as well as injection mould thickness and standard mould constraint.

7.4.4.1.1 Injection mould plate length and width constraint

The suitable length and width of the mould plates is constrained by the number of cavities, their layout and the space between the cavities and the side of the mould (see

figure 7.35). The latter is referred to as the variable "s" and is determined using table 7.2 (Menges, Michaeli et al. 2001).

Table 7.2 Minimum space required in an injection mould between the cavity and the side of the mould

Part length, width or depth	Minimum space between cavity and mould (s)
0 – 25 mm	5 mm
26 – 50 mm	10 mm
51 – 75 mm	20 mm
76 – 100 mm	30 mm
101 – 125 mm	40 mm
126 – 150 mm	55 mm
151 – 175 mm	75 mm
176 – 200 mm	100 mm
201 – 225 mm	125 mm
226 – 250 mm	150 mm

As shown in figure 7.35, the formulas for calculating a suitable length and width of the mould plates were captured in a rule based format as recommendation rules. After these rules were captured, they were formally represented as methods of the "Injection Mould Plate Length Constraint" and the "Injection Mould Plate Width Constraint" class (see figure 7.36).

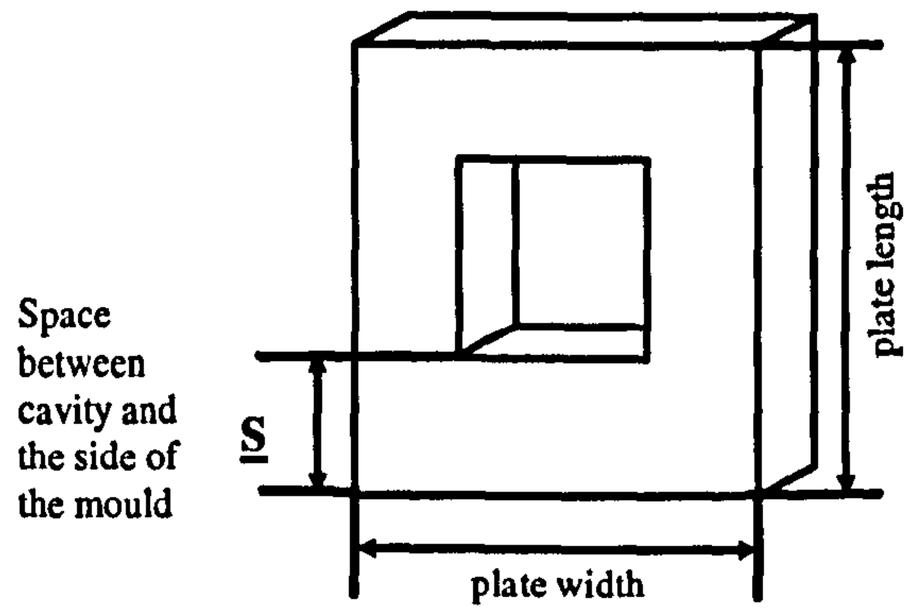


Plate width constraint

if (type arrangement = unbalanced 1) then
if (part width > part length)

$$plate\ width = (4 \times s) + (2 \times part\ length)$$

else

$$plate\ width = (4 \times s) + (2 \times part\ width)$$

if (type arrangement = unbalanced 2) then

$$plate\ width = \left(\left(\frac{num\ cavities}{4} + 2\right) \times s\right) + \left(\frac{num\ cavities}{4} \times part\ width\right)$$

if (type arrangement = balanced 1) then

$$plate\ width = (4 \times s) + (2 \times part\ width)$$

if (type arrangement = balanced 2) then
if (par twidth > part length)

$$plate\ width = \left(\left(\frac{num\ cavities}{4} + 1\right) \times 2 \times s\right) + \left(\frac{num\ cavities}{2} \times part\ length\right)$$

else

$$plate\ width = \left(\left(\frac{num\ cavities}{4} + 1\right) \times 2 \times s\right) + \left(\frac{num\ cavities}{2} \times part\ width\right)$$

Plate length constraint

if (type arrangement = unbalanced 1) then
if (part width > part length)

$$plate\ length = \left(\left(\frac{num\ cavities}{2} + 2\right) \times s\right) + \left(\frac{num\ cavities}{2} \times part\ width\right)$$

else

$$plate\ length = \left(\left(\frac{num\ cavities}{2} + 2\right) \times s\right) + \left(\frac{num\ cavities}{2} \times part\ length\right)$$

if (type arrangement = unbalanced 2) then

$$plate\ length = \left(\left(\frac{num\ cavities}{4} + 2\right) \times s\right) + \left(\frac{num\ cavities}{4} \times part\ length\right)$$

if (type arrangement = balanced 1) then

$$plate\ length = (4 \times s) + (2 \times part\ length)$$

if (type arrangement = balanced 2) then
if (part width > part length)

$$plate\ length = (4 \times s) + (2 \times part\ width)$$

else

$$plate\ length = (4 \times s) + (2 \times part\ length)$$

Figure 7.35 Injection mould plates design constraints

Injection Mould Plate Width Constraint
+ suggestValue(String cav arrangement, int cavities, float part width, float part length) : double *if (cav arrangement = unbalanced 1)* *if (part width > part length)* *calculateS()* $plate\ width = (4 \times s) + (2 \times part\ length)$ *else* *calculateS()* $plate\ width = (4 \times s) + (2 \times part\ width)$ *if (cav arrangement = unbalanced 2)* *calculateS()* $plate\ width = \left(\left(\frac{cavities}{4} + 2\right) \times s\right) + \left(\frac{cavities}{4} \times part\ width\right)$ *if (cav arrangement = balanced 1)* *calculateS()* $plate\ width = (4 \times s) + (2 \times part\ width)$ *if (cav arrangement = balanced 2)* *if (part width > part length)* *calculateS()* $plate\ width = \left(\left(\frac{num\ cavities}{4} + 1\right) \times 2 \times s\right) + \left(\frac{num\ cavities}{2} \times part\ length\right)$ *else* *calculateS()* $plate\ width = \left(\left(\frac{num\ cavities}{4} + 1\right) \times 2 \times s\right) + \left(\frac{num\ cavities}{2} \times part\ width\right)$

Injection Mould Plate Length Constraint
+ suggestValue(String cav arrangement, int cavities, float part width, float part length) : double *if (cav arrangement = unbalanced 1)* *if (part width > part length)* *calculateS()* $platelength = \left(\left(\frac{numcavities}{2} + 2\right) \times s\right) + \left(\frac{numcavities}{2} \times part\ width\right)$ *else* *calculateS()* $platelength = \left(\left(\frac{numcavities}{2} + 2\right) \times s\right) + \left(\frac{numcavities}{2} \times part\ length\right)$ *if (cav arrangement = unbalanced 2)* *calculateS()* $platelength = \left(\left(\frac{numcavities}{4} + 2\right) \times s\right) + \left(\frac{numcavities}{4} \times part\ length\right)$ *if (cav arrangement = balanced 1)* *calculateS()* $plate\ length = (4 \times s) + (2 \times part\ length)$ *if (cav arrangement = balanced 2)* *if (part width > part length)* *calculateS()* $plate\ length = (4 \times s) + (2 \times part\ width)$ *else* *calculateS()* $plate\ length = (4 \times s) + (2 \times part\ length)$

Figure 7.36 "Injection Mould Plate Width Constraint" and "Injection Mould Plate Length Constraint" class representation using UML notation

7.4.4.2 Injection mould thickness constraint

The suitable thickness of the mould is also constrained by the minimum space between the cavities and the side of the mould (variable s), as shown in the following formula:

$$mould\ thickness = (2 \times s) + product\ depth$$

This rule was formally represented as a method of the "Injection Mould Thickness Constraint" class (see figure 7.37).

Injection Mould Thickness Constraint
+ suggestValue(String cav arrangement, float part depth) : double *calculateS()* *mould thickness* = $(2 \times s)$ + *product depth*

Figure 7.37 "Injection Mould Thickness Constraint" class representation using UML notation

7.4.4.3 Standard mould constraint

A standard mould unit is generally required to fabricate an insert/bolster mould. In order to enable the selection of a suitable standard mould unit, a list of commercial units capacities should be made available. Figure 7.38 illustrates the Engineering Data Model, which is referred to as Standard Mould Model. In this model, the "Standard Mould" class contains as attributes the standard mould units' capacities, such as width, length, minimum and maximum mould thickness.

Standard Mould
- name: String - length: float - width: float - minimum mould thickness: float - maximum mould thickness: float - minimum cavity thickness: float - maximum cavity thickness: float - minimum core thickness: float - maximum core thickness: float

Figure 7.38 Standard Mould Model representation using UML notation

In order to suggest the best standard mould to use, the following formula was captured as a recommendation rule:

$$(std\ mould\ length \geq plate\ length)$$
and
$$(std\ mould\ width \geq plate\ width)$$
and
$$\left(\frac{std\ mould\ min\ mould\ thickness + std\ mould\ max\ mould\ thickness}{2} \geq mould\ thickness\right)$$

This rule is formally represented in the method "suggestStdMould" of the "Standard Mould Constraint" class shown in figure 7.39.

Standard Mould Constraint
+ suggestStdMould(Collection std moulds, float plate length, float plate width, float mould thickness) : String *for all (std moulds)* *if (length suggested >= std mould length >= part length) and* *(width suggested >= std mould width >= part width) and* $\left(thickness\ suggested >= \frac{std\ mould\ min\ thickness + std\ mould\ max\ thickness}{2} >= mould\ thickness\right)$ *std mould = std mould name* *length suggested = std mould length* *width suggested = std mould width* $thickness\ suggested = \frac{std\ mould\ min\ thickness + std\ mould\ max\ thickness}{2}$ *"The recommended standard mould is (std mould name) with a length of (length), width of (width) and an average thickness of (thickness suggested)"*

Figure 7.39 "Standard Mould Constraint" class representation using UML notation

After all the manufacturing constraints of the mould plates were formally represented using UML notation, they were represented as aggregation relationships of the "Injection Mould Design Constraints" class shown in figure 7.40.

The procedure followed to identify, capture and represent the design knowledge for the components of the mould is the same than that followed for the mould plates. Therefore, the following subsections present the identified and captured knowledge. The representation of these constraints in UML notation is shown in detail in Appendix E.

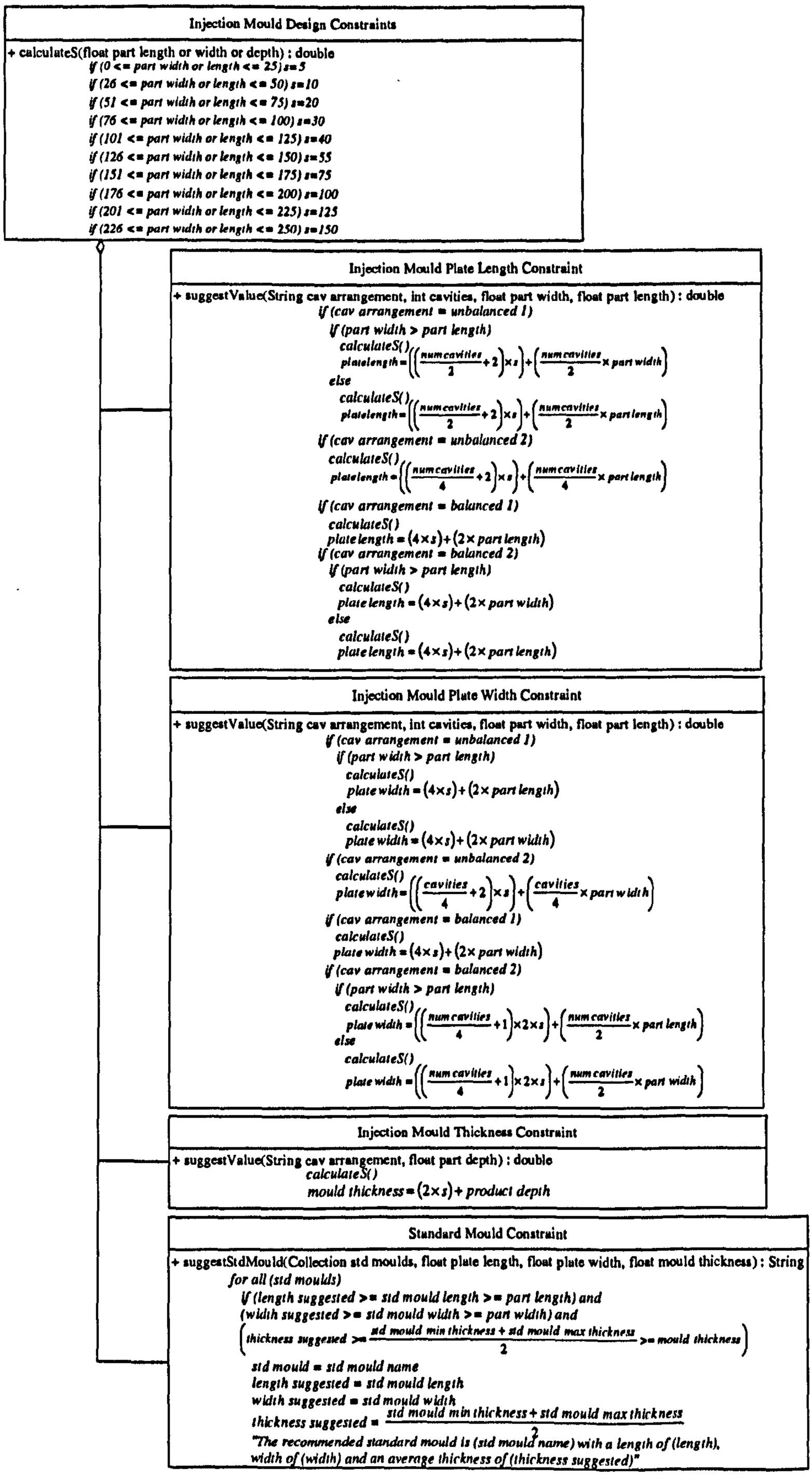


Figure 7.40 Representation of the "Injection Mould Design Constraints" class and its relationships using UML notation

7.4.4.2 Modelling of the cooling system design constraints

The mould must be cooled to remove heat from the moulded part so it can be ejected from the mould as quickly as possible. Cooling is fabricated by drilling or machining passages in the mould and circulating a heat transfer fluid through those passages. The cooling passages in the mould are normally interconnected to form a circuit. The circuit may be at one or more levels, the number of which will depend on the depth of the mould plate (Dow 2001).

Some of the constraints when designing the cooling circuit are (Pye 1989; Dow 2001):

- As figure 7.41-a illustrates, the passages must not be positioned too close to the cavity (closer than 16 mm), as this causes temperature variations across the cavity.
- The cooling passages must not be positioned near probable weld lines areas because these require less cooling in order to have better welding.
- The cooling passages must be drilled to accept pipes in the range of 6 to 10mm of diameter (see figure 7.41-b). Smaller pipes must not be used unless there is a size constraint.
- The layout of the circuit is often complicated by the fact that flow ways must not be drilled too close to any other hole in the same mould plate. Other holes accommodate ejector pins, guide pillars, sprue, inserts, etc. To obtain the best possible position it is good practice to lay the circuit in at the earliest opportunity in the design.
- The circuits for cavity and core plates are generally dissimilar. Figure 7.41-d shows the considerations that should be taken into account in order to select the most suitable circuit type. Some of these considerations are (Pye 1989):

<u>For integer type cavity plates:</u>

- U-circuit is useful for cooling thin and small cavities.
- The Z-configuration is suitable for a large area, shallow cavity and can be positioned directly below the cavity.
- The multi level system is suitable for cooling deep cavities.

- Coolant plates should be used when it is required to control the temperature of the individual walls of the cavity.

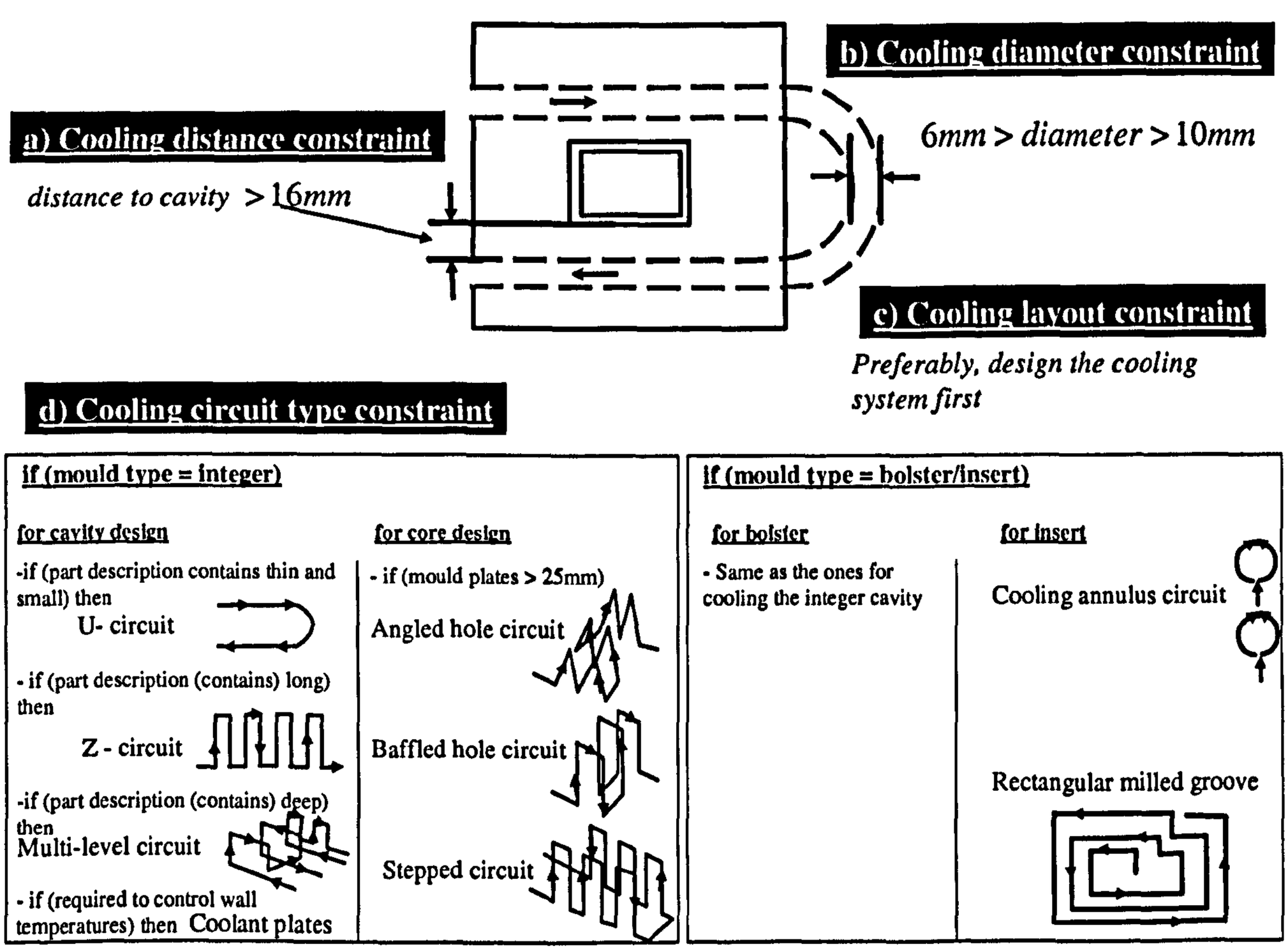


Figure 7.41 Capturing cooling system design constraints

For integer type core plates:

- For plates deeper than 25 mm an angled hole system, baffled straight hole system or stepped circuit could be used. These systems can be used for fairly narrow cores. For wider cores a number of identical circuits can be incorporated.

For bolster:

- The methods for cooling the bolster are identical to the ones for cooling the integer cavity plate.
- It is desirable that the cooling channels are positioned as close to the insert as practicable.

For insert:

- Circular shaped inserts can incorporate coolant annulus circuits by machining the coolant annulus into the periphery of the insert.
- For rectangular shapes, the annulus can also be incorporated as a groove machined. The main object of this approach is to layout the cavities so that the individual grooves interconnect and thereby avoid the necessity of additional drilled holes.

The identified manufacturing constrains highlighted an interaction between the cooling systems and the other components of the mould because the circuit type should be firstly considered. Another interaction was highlighted between the cooling system and the features of the part. This is because the manufacturability of the part's features must be ensured before considering the circuit type and position of the cooling system. Otherwise, problems would still arise even when the cooling systems has been designed correctly. To formally represent these interactions in UML notation, the manufacturing constraints representations of both the part features and the cooling system were linked using a dotter arrow. The same approach is used for the interactions of the other components' manufacturing constraints.

7.4.4.3 Modelling of the ejection system design constraints

The ejection system is a mechanical component of the mould that provides a mechanism by which the moulded part can be positively ejected from the core. This component must be considered when designing the injection moulded part to avoid expensive mould costs like side cores, or external inserts (Dieter 2000).

The design of the ejection system includes designing the ejector plate assembly and selecting the suitable ejection method (Pye 1989). The tool designer has several ejection techniques from which to choose, such as pin ejection, stepped pin, sleeve ejection, blade ejection, valve ejection, stripper bar ejection, stripper plate ejection and air ejection. Several factors must be considered when selecting and designing the suitable ejection

system: the shape of the part, part dimensions, type of features on the part and walls' thickness. Some of the guidelines are the following (Pye 1989; Dow 2001):

- Pin ejectors are very common and inexpensive methods. This type is the standard type of ejector used for ejecting most types of box shaped parts. The pins should be arranged to push on the bottom of the side walls of the part. Corners and ribs also provide rigid structure from which to eject the part. The working diameter of the ejector pin must be a good slide fit in its mating hole in the mould plate and must be 5 mm or 10 mm more than the wall thickness of the part. The recommended thickness for the head of the pin is 3 mm.
- Stepped pins are used where changes in shape occur (at ribs, bosses), because these features increase the difficulty of the ejection. For a stepped pin the diameter must be less than 3 mm. The recommended thickness for the head of the pin is 3 mm.
- Sleeve ejectors are often used for certain types of circular parts or circular bosses on a part of any shape. This method is particularly efficient because the ejection force is applied to a relatively large surface area.
- Blade ejection is a rectangular ejector pin and is suitable for ejection of very slender parts, such as ribs and other projections.
- Valve ejection is a large diameter ejector pin and is normally used for the ejection of relatively large parts in situations where it is impracticable to use standard surface pins.
- Stripper bar ejection is used with box and thin walled parts. As a single bar is used for ejection, the marks left on the surface of the part are reduced to a minimum.
- Stripper plate ejection is used primarily for the ejection of the rotational box mouldings.
- Air ejector is used to eject parts having an "enclosed" geometry. It can also be used for large mouldings such as body parts for the automotive industry. The ejector force is provided by compressed air which is introduced via a small air ejector.

Regardless of the ejection method selected, the part surface where the pins are going to be located should be carefully chosen with the collaboration of the product engineer. If the surface area of ejection is inadequate, the part surface can be damaged by the ejection mechanism.

The identified manufacturing constraints for the ejection system highlighted an interaction between the ejection system and the part features. This is because the manufacturability of the features should be ensured before considering what type of ejection system is the most suitable to use.

7.4.4.4 Modelling of the venting system design constraints

Prior to melt plastic material being injected into a mould, the cavity is occupied by the room air that was trapped when the mould was closed. As the melt enters the cavity, the air must have a mean to escape (Osswald, Turng et al. 2002). It is good practice to provide vents in the mould to release the air that is displaced when the plastic flows into it. In multi cavity mould vents should be placed in the runners so that air cannot get into the cavity (Menges, Michaeli et al. 2001). Poor venting can results in short shots, weak welding lines, burn marks and stress resulting from high injection pressure (Dow 2001). The following constraints must be taken into account when designing the vents in the mould (Pye 1989; Dow 2001; Osswald, Turng et al. 2002):

- Positions where a vent is likely to be required are:
    - Vents should be placed at the last place in the mould expected to be filled, which is at the point furthermost from the gate on symmetrical cavities.
    - At the point where flow paths are likely to meet.
    - Anywhere along the parting line of the mould. A reasonable guide is to have vents spaced at 25 mm pitch.
    - For ribs and bosses, vents may be incorporated into the mould by incorporating an ejector pin in the required position. The minute gap between the ejector pin and the mould plate hole is sufficient to allow air to escape.
- Vents normally consist of two regions, the vent land and vent relief. The vent land at the perimeter of the cavity must be very shallow so that air can escape, but not the molten plastic. The depth of the vent must be in the range of 0.02 mm to 0.05 mm for at least the first 2 mm distance from the edge of the mould cavity. The depth and

length of the vent relief is not as critical, but is commonly about 0.4 mm deep. The vent width must be a minimum of 3 mm.

These manufacturing constraints highlighted interactions between the venting system and the gating system as well as the ejection system. This interaction is because the venting position depends on the positions of the gate(s), and therefore the later should be firstly designed within manufacturing constraints. The ejection mechanism should also be defined before the venting system because the first can act as a vent if it is positioned on ribs and bosses.

7.4.4.5. Modelling of the feed system design constraints

The feed system is the flow way in the mould, which connects the nozzle of the injection machine to each cavity. It is desirable to keep the distance that the material has to travel down to a minimum to reduce pressure and losses. It is for this reason that careful consideration must be given to the cavity layout. The feed system comprises the runners, the gates and the sprue. The following subsections explore the design constraints for each of these elements of the mould.

7.4.4.5.1 Modelling of the runner system design constraints

The runner is a channel machined into the mould plate to convey the molten plastic material from the sprue to the entrance (gate) of the cavity. The following considerations should be taken into account when designing the runner system (Pye 1989; Al-Ashaab 1994; Dow 2001):

- Cross sectional shape: The runner should provide a maximum cross sectional area from the standpoint of pressure transfer and minimum contact on the periphery from the standpoint of heat transfer. The shape of the cross section of the runner can be (Pye 1989; Dow 2001):

1. Round runner: this type of runner is the most efficient one because the gate is positioned in line with the centre of the runner to receive the material from the central flow stream (see figure 7.42-a). However, it also is more expensive to

provide, because the runner must be cut into both halves of the mould. This type of runner should be used for simple two plate moulds.

2. Trapezoidal runner: This type of runner is yet efficient and not too expensive. An angle of 10 degrees per side must be incorporated on the runner wall, with the depth of the trapezoid equal to its width (see figure 7.42-b). This type of runner should be used for moulds that have a complex parting surface.
3. Semicircular runner: This type has been developed to improve the trapezoidal, and the same constraints apply to it (see figure 7.42-c).

- The size of the runner: The diameter of the runner must not be below 2 mm nor above 10 mm to avoid early freeze off or excessive cycle time. A formula used to calculate the runner diameter is (Pye 1989):

$$D = \frac{\sqrt{W} \times \sqrt[4]{L}}{3.7}$$

$D = runner\ diameter$

$W = weight\ of\ moulding$

$L = height\ of\ runner$

- The runner layout: In order to select a runner layout is necessary to consider the number of cavities, the shape of the part, the type of the mould and the type of gate. As explained in the mould plates constraints in section 7.4.4.1, the cavity layout can be balanced or unbalanced. A balanced arrangement ensures that all the cavities will fill uniformly and without interruption providing the gate land lengths and areas are identical. However, this is unpractical for moulds which incorporate a large number of differently shaped cavities.

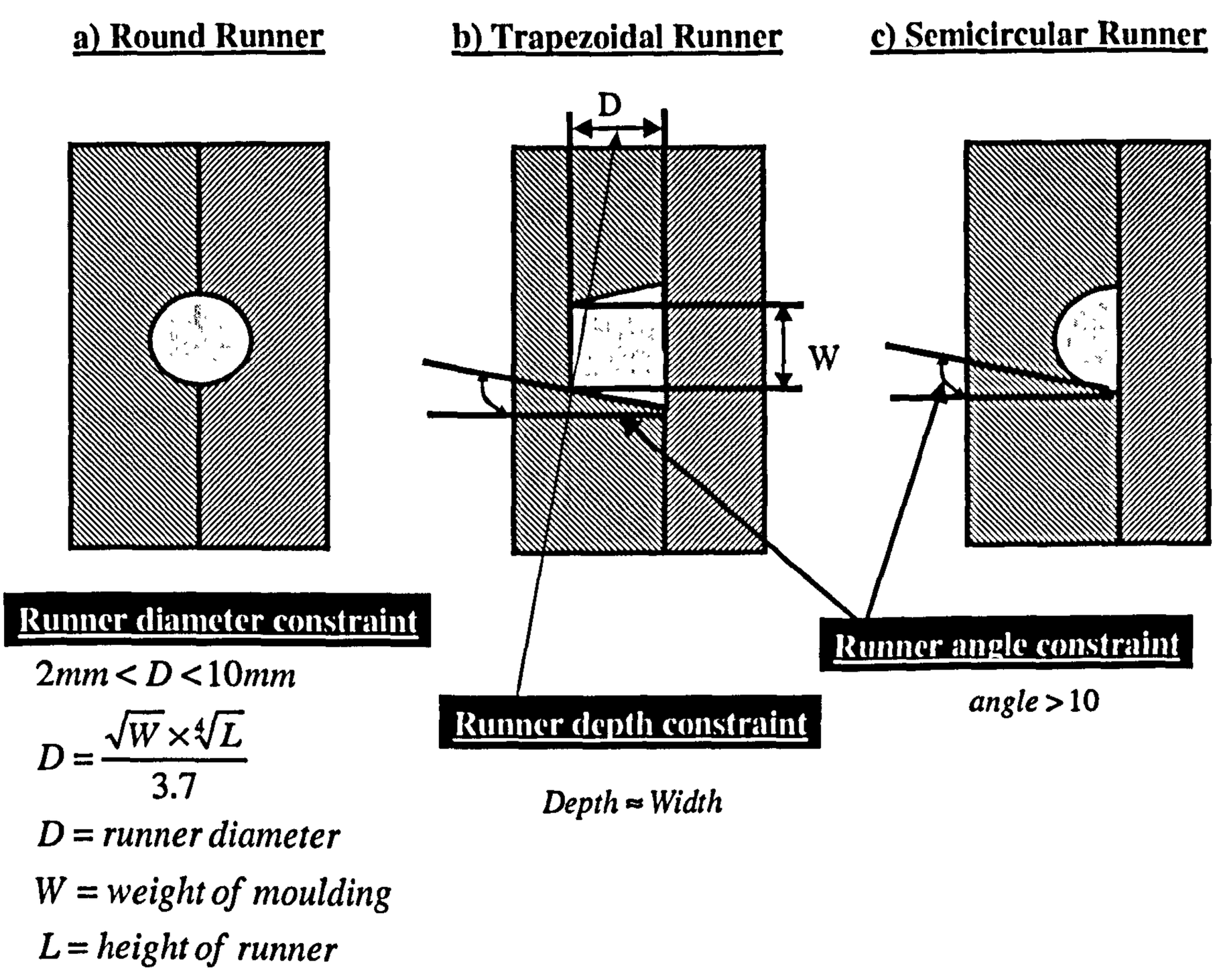


Figure 7.42 Capturing runner system design constraints

The identified manufacturing constraints for the runner system highlighted an interaction with the features in the part. This is because the manufacturability of the features should be ensured before considering the design of the runner. The same interaction was highlighted for the other components of the feed system (gating system and sprue).

7.4.4.5.2 Modelling of the gating system design constraints

The gate is a channel or orifice connecting the runner with the cavity, and should be designed to permit easy filling of the mould. It has a small cross sectional area when compared with the rest of the feed system. The constraints for the design of the gates are (Pye 1989; Dieter 2000; Dow 2001; Osswald, Turng et al. 2002):

- Gates should be small enough to ensure easy separation of the runner and the part. However, they should be large enough to prevent premature freezing off of the plastic flow, which can affect consistency of the part dimensions. The minimum size

suggested for the gate diameter is 0.75 mm, and as a rule, it must not exceed the runner or sprue diameter (see figure 7.43-a).

- The location of the gates for entry of the plastic material into the die is a crucial design detail. Close collaboration between the product engineer and tool designer is required to select the best location of the gates. Some considerations that can help in this process are (see figure 7.43-b):
  - Gates should be position to ensure an even flow of melt in the cavity, so that it fills uniformly and the advancing melt front spreads out and reaches the various cavities extremities at the same time. For rotational parts this position is at the centre of the base. For prismatic parts the central position is also preferred. However, weld lines are to be expected when two flows meet. This defect can be prevented by keeping the cooling medium away from the neighbourhood of weld lines.
  - In order to minimize weld lines, gates should be positioned at the farthest distance from an obstruction.
  - Gate should be positioned in thickest wall section if there is a variation of wall thickness in the product. Gating into thinner walls will restrict the control of the packaging of the thicker region. This can result in excessive shrinkage, warpage, sinks and voids.
  - Gates should be positioned in line with the centre of the runner to receive the material from the central flow stream.
- To obtain the optimum filling conditions, the type of gate must be carefully chosen according to the part characteristics. The types of gates commonly used are: sprue gate, edge gate, pin gate, tab gate, diaphragm gate, ring gate, film gate (Cracknell and Dyson 1993; Menges, Michaeli et al. 2001; Osswald, Turng et al. 2002). The following subsections will describe three of them in more detail.

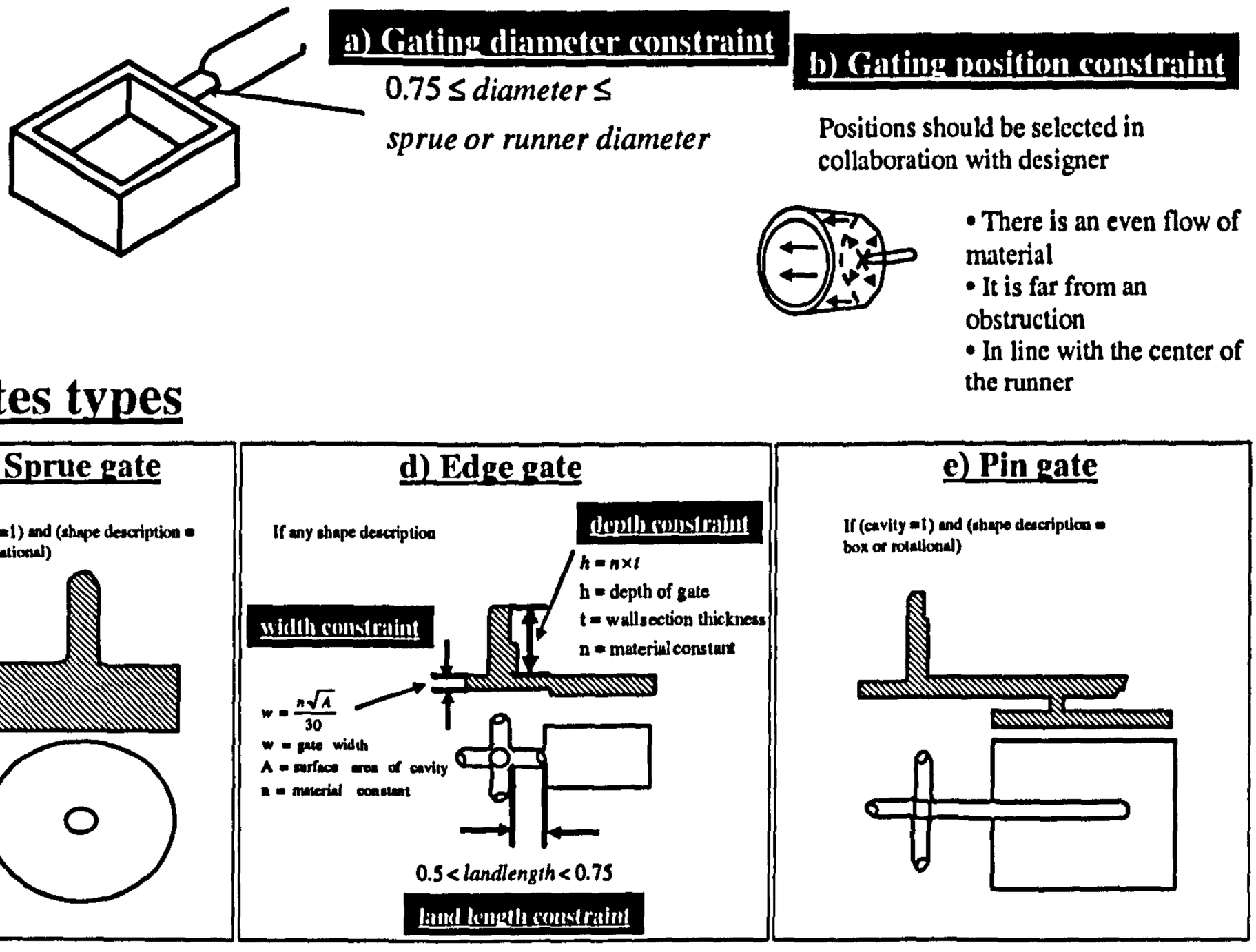


Figure 7.43 Capturing gating system design constraints

7.4.4.5.2.1 Sprue gate

This gate is recommended for single cavity moulds or moulds for circular parts requiring symmetrical filling (see figure 7.43-c). This gate is also suitable for thick walls (Dow 2001). The main disadvantage with this type of gate is that it leaves a large gate mark on the part. This mark can be reduced if the dimensions of the gate, specially the length, are reduced (Pye 1989).

7.4.4.5.2.2 Edge gate

This is a general purpose gate and its simplest form is merely a rectangular channel machined in one mould plate to connect the runner to the cavity (see figure 7.43-d). This type of gate is used for multicavity two plates moulds and is suitable for medium and thick walls (Dow 2001). Because the rectangular shape of the gate, the dimensions are given by width, depth and land length (Pye 1989). As figure 7.43-d illustrates, the land

length must be kept within a range from 0.5 mm to 0.75 mm, keeping it as small as possible. The gate depth can be calculated using the wall thickness of the part as follows:

$$h = n \times t$$

h = depth of gate mm

t = wall thickness

n = material constant

In addition the gate width can be calculated as follows:

$$w = \frac{n\sqrt{A}}{30}$$

w = gate width

A = surface area of cavity

n = material constant

7.4.4.5.2.3 Pin gate

This is a circular gate used for feeding into the base of the part and because it is relatively small in diameter it is often preferred to the sprue gate (Pye 1989). This type of gate, shown in figure 7.43-e, is good for applications that require automatic degating or multi point feeding, but is suitable only for thin walls (Dow 2001). The gate dimensions that should be considered are land length, and gate diameter. The land length constraint is the same as for the edge gate.

7.4.4.5.3 Modelling of the sprue design constraints

The sprue connects the nozzle of the injection machine to the runner system of the mould. Ideally, the sprue should be as short as possible to minimize material and cycle time. The design constraints are shown in figure 7.44 (Menges, Michaeli et al. 2001).

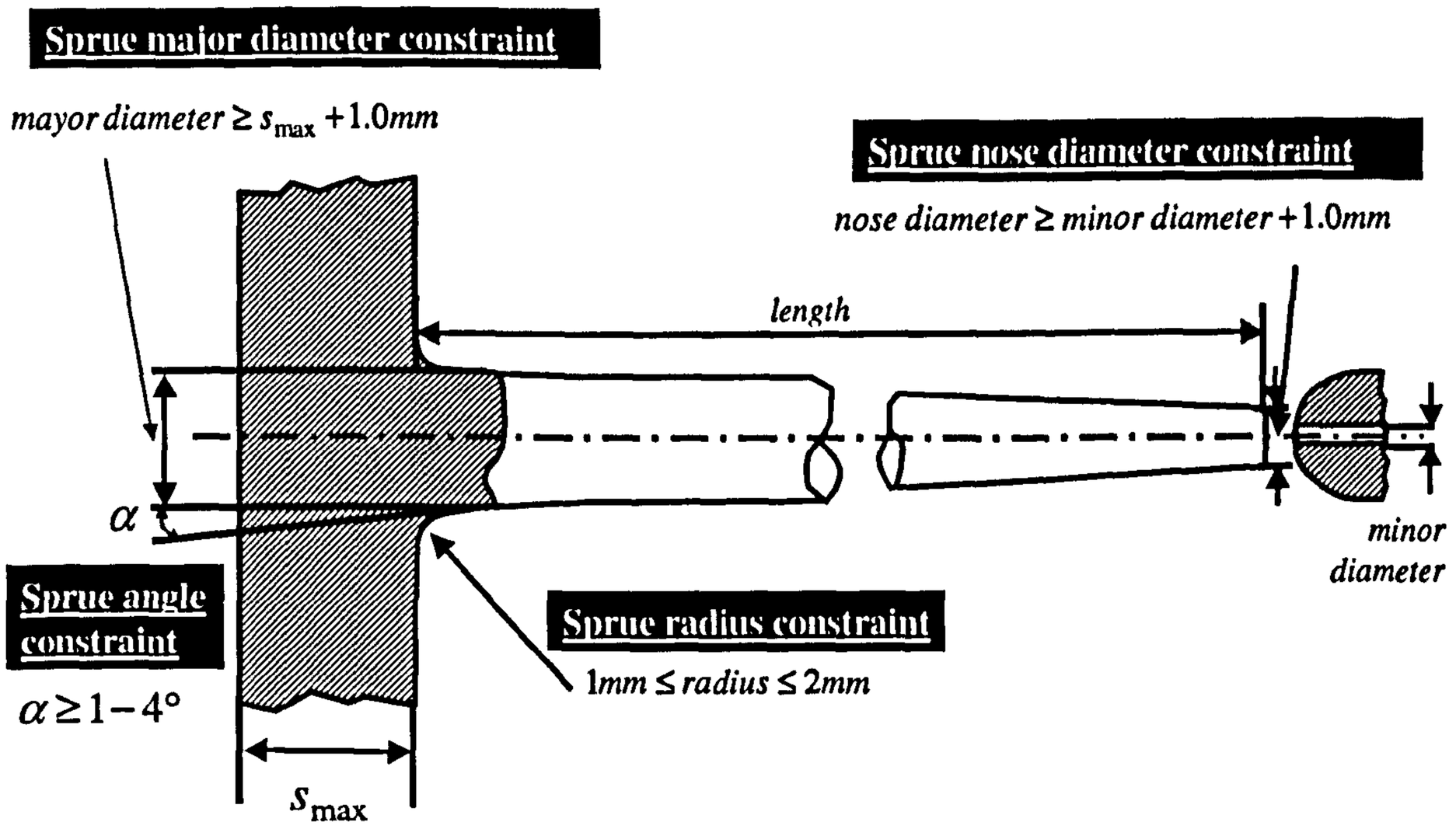


Figure 7.44 Capturing main sprue design constraints

7.4.4.6 Injection mould design constraints formal representation

After the manufacturing constraints for the different mould components were represented in UML classes, they were grouped using UML packages. Figure 7.45 illustrates the packages: "Cooling Systems Design Constraints", "Ejection System Design Constraints", "Feed System Design Constraints" and "Venting System Design Constraints". These packages have an aggregation relationship with the "Injection Mould Design Constraints" class.

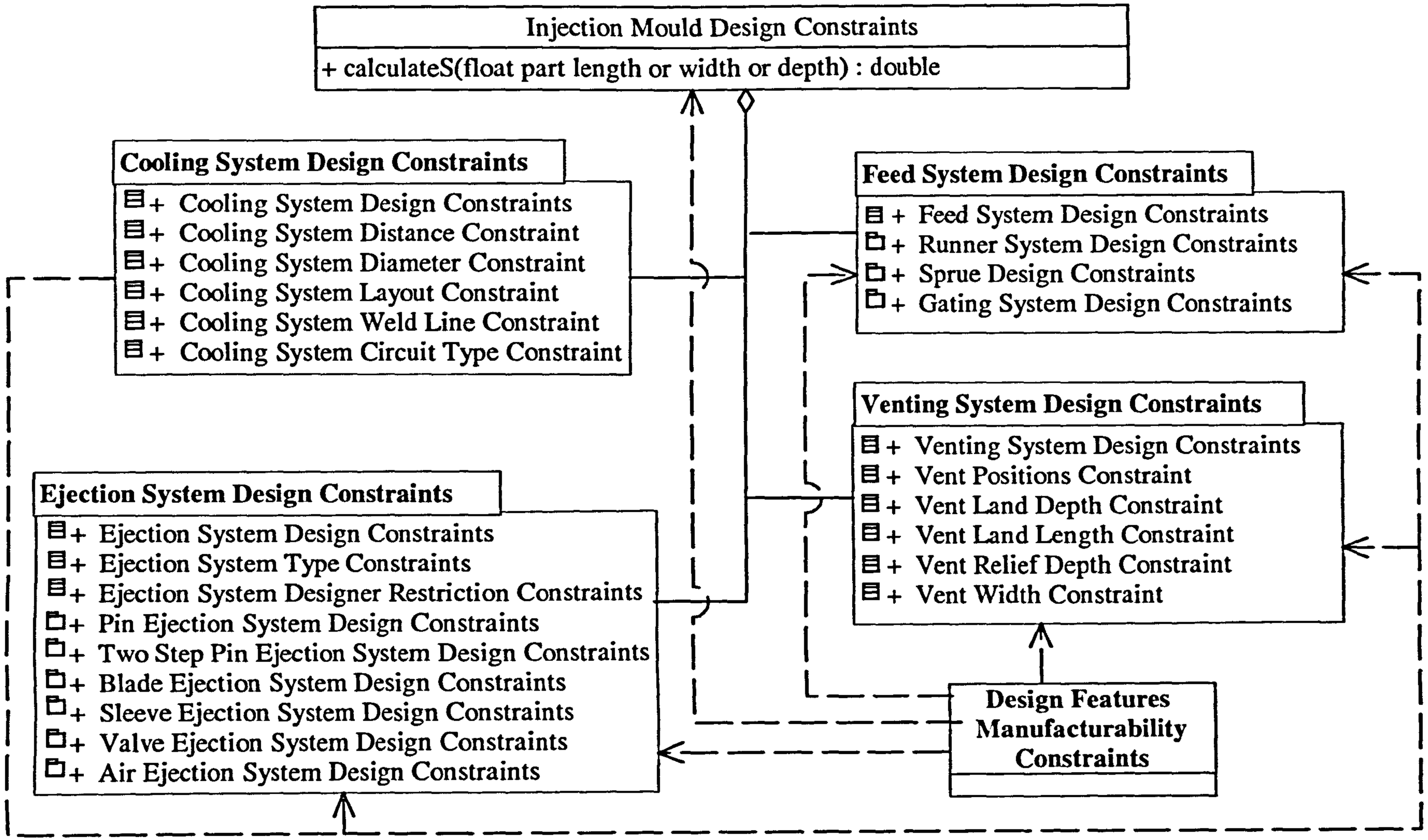


Figure 7.45 Representation of the "Injection Mould Design Constraints" class and its relationships using UML notation

Furthermore, the interactions among the manufacturing constraints were represented using a dotted arrow connecting the different classes. These interactions are not only among manufacturing constraints for the mould design but also interactions with the design features manufacturing constraints.

7.4.5 Modelling of the injection mould fabrication constraints

Injection moulds are fabricated using different manufacturing process. Hence the knowledge required to support the mould fabrication depends on the capabilities of the manufacturing process used. For example, considerations to determine the machining process plan depend on the machining capabilities. These considerations are not covered in this work as it is out of scope exploring other processes capabilities. In this research, only the considerations for selecting the best manufacturing process according to the part data is addressed. For this, the mould was broken up in core, cavity and other systems. The constraints imposed on these components were then identified, captured and represented as presented in the following subsections.

7.4.5.1 Injection mould core and cavity fabrication constraints

The suitable method required for fabricating the cavity and the core depends upon whether the form is of a simple or a complex nature. Simple forms can be produced by conventional tools such as lathe, grinding, shaping and milling machine. However, complex shapes require of other techniques, which include investment casting, electro deposition, cold hobbing, pressure casting, laser carving and spark machining (Pye 1989). This rule was formally represented in an object oriented class called "Cavity Fabrication Technique" class. As shown in figure 7.46, this class interacts with the "Design Features Manufacturability Constraints" class, as the part features needs to be within limitations before being considered. The same rule is captured for the core and it is stored in a class named "Core Fabrication Technique Constraint".

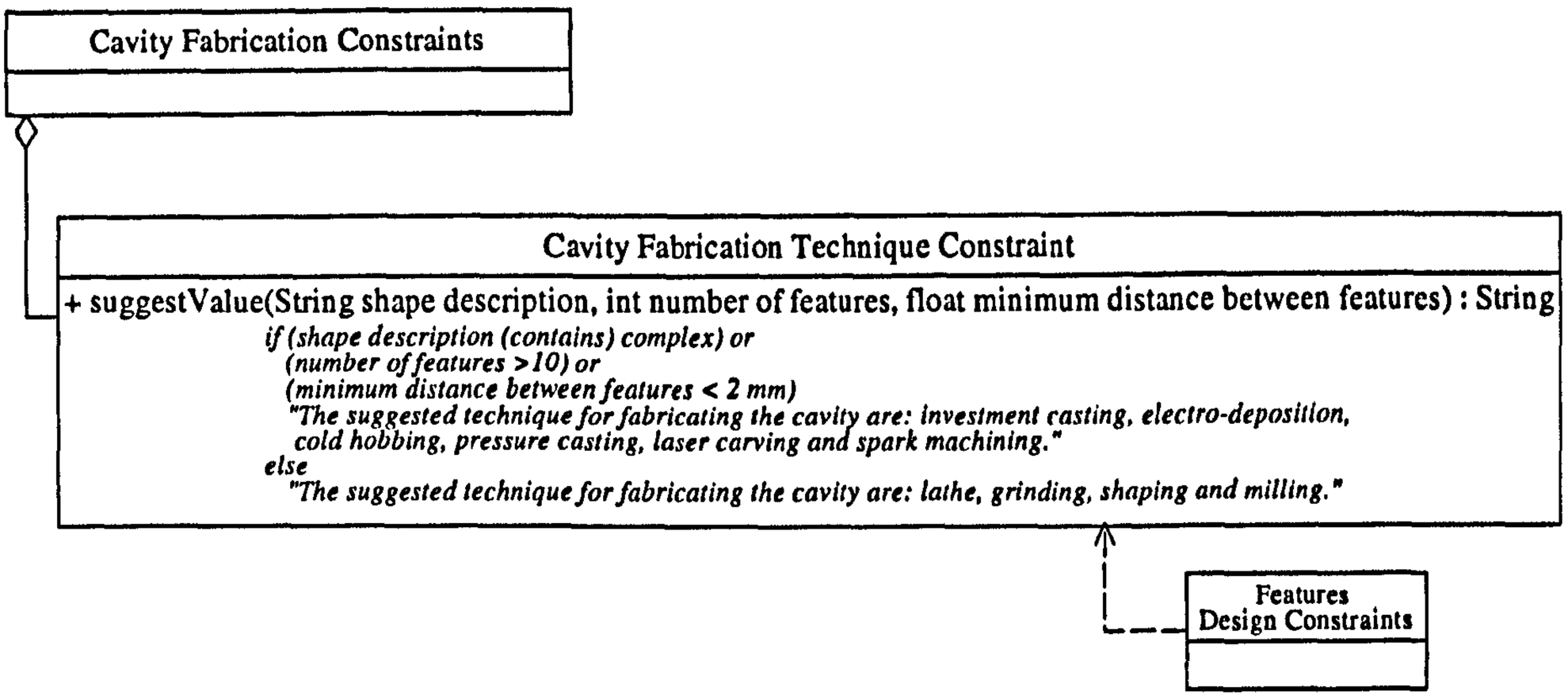


Figure 7.46 "Cavity Fabrication Constraints" class representation using UML notation

The different components of the mould placed in the core and cavity, such as cooling system, venting system, ejection system and feed system, are commonly produced on conventional machine tools, such as lathe, grinding, shaping and milling machine. This is because its machining is not as critical as the manufacture of the cavity and core forms but, nevertheless, accuracy in the manufacture of the various parts is necessary to ensure that the mould can be assembly by the fitter without an excessive amount of bench work (Pye 1989). This constraint is captured in the class shown in figure 7.47.

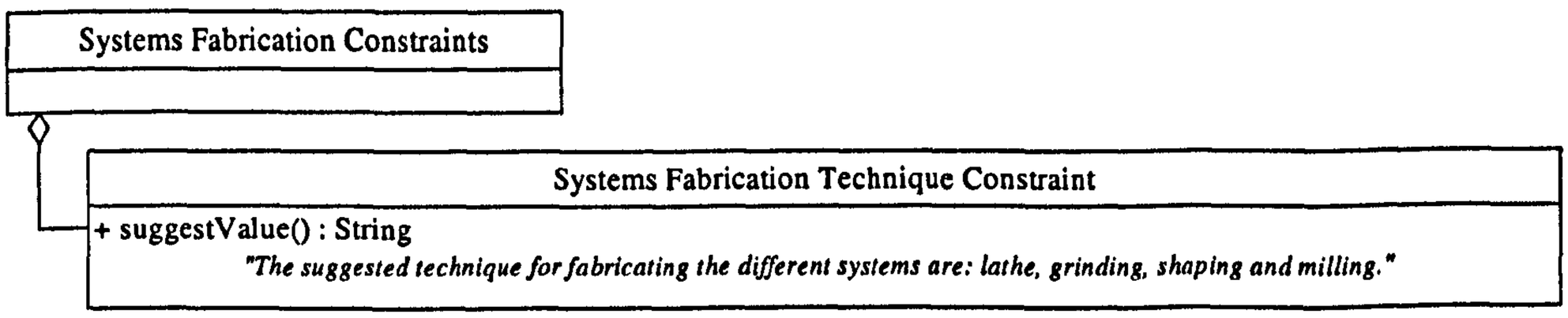


Figure 7.47 "Systems Fabrication Constraints" class representation using UML notation

7.4.5.2 Injection mould fabrication constraints formal representation

In the Manufacturing Knowledge Model, the considerations for the mould fabrication were represented in classes and grouped in packages. These packages are: "Cavity Fabrication Constraints", "Core fabrication Constraints" and " Systems Fabrication

Constraints". They were linked using an aggregation relationship with the "Mould Fabrication Constraints" class, as shown in figure 7.48.

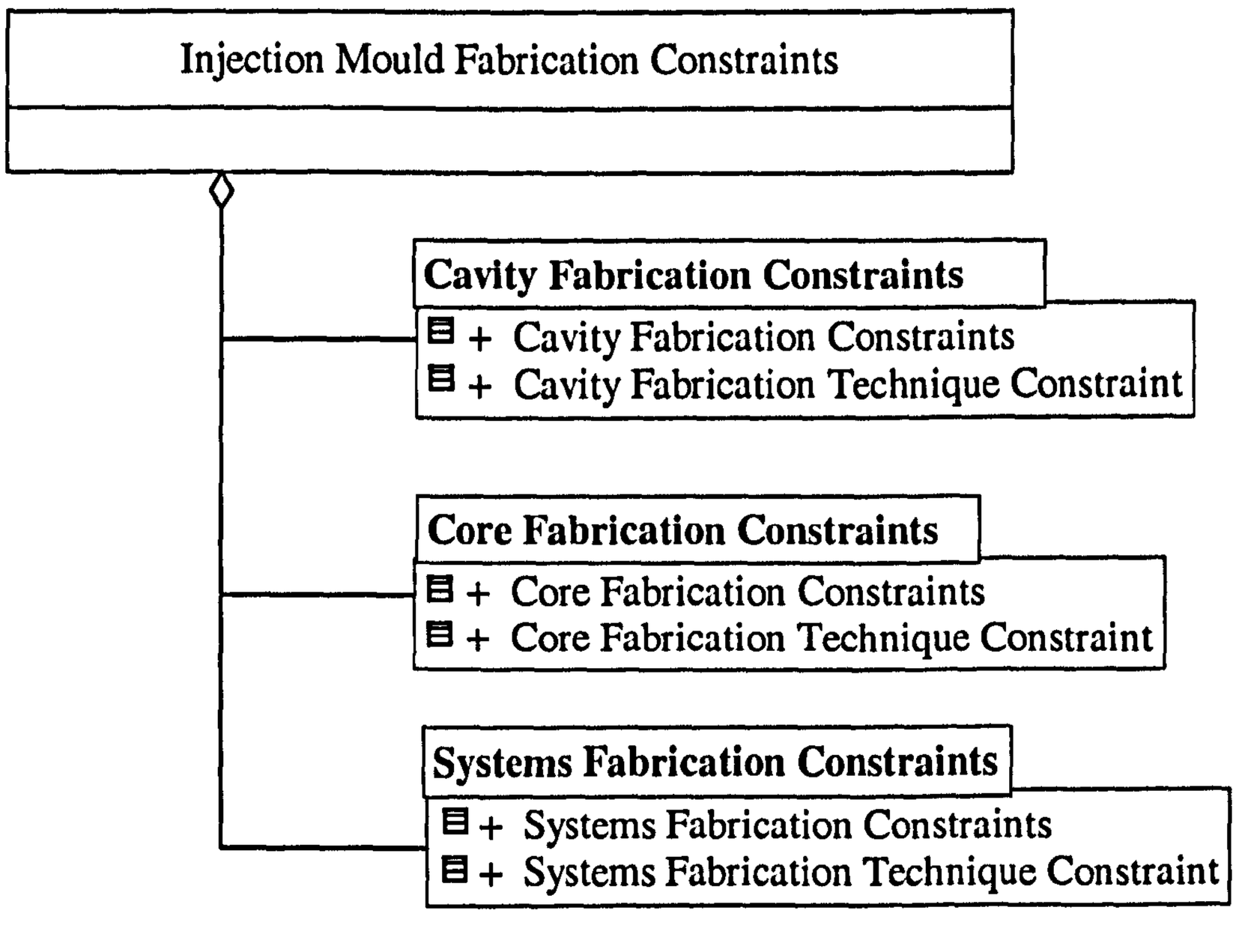


Figure 7.48 Representation of the "Injection Mould Fabrication Constraints" class and its relationships using UML notation

7.5 Manufacturing Knowledge Model formal representation

The manufacturing constraints and their interactions, explained in the previous sections, were brought together in the Manufacturing Knowledge Model. Figure 7.49 shows the top view of this model, where the manufacturing constraints for the different activities are geographically distributed among the collaborators. In the Manufacturing Knowledge Model a company has been represented as an object, and its knowledge is represented through an aggregation relationship with the class called "Manufacturing Constraints".

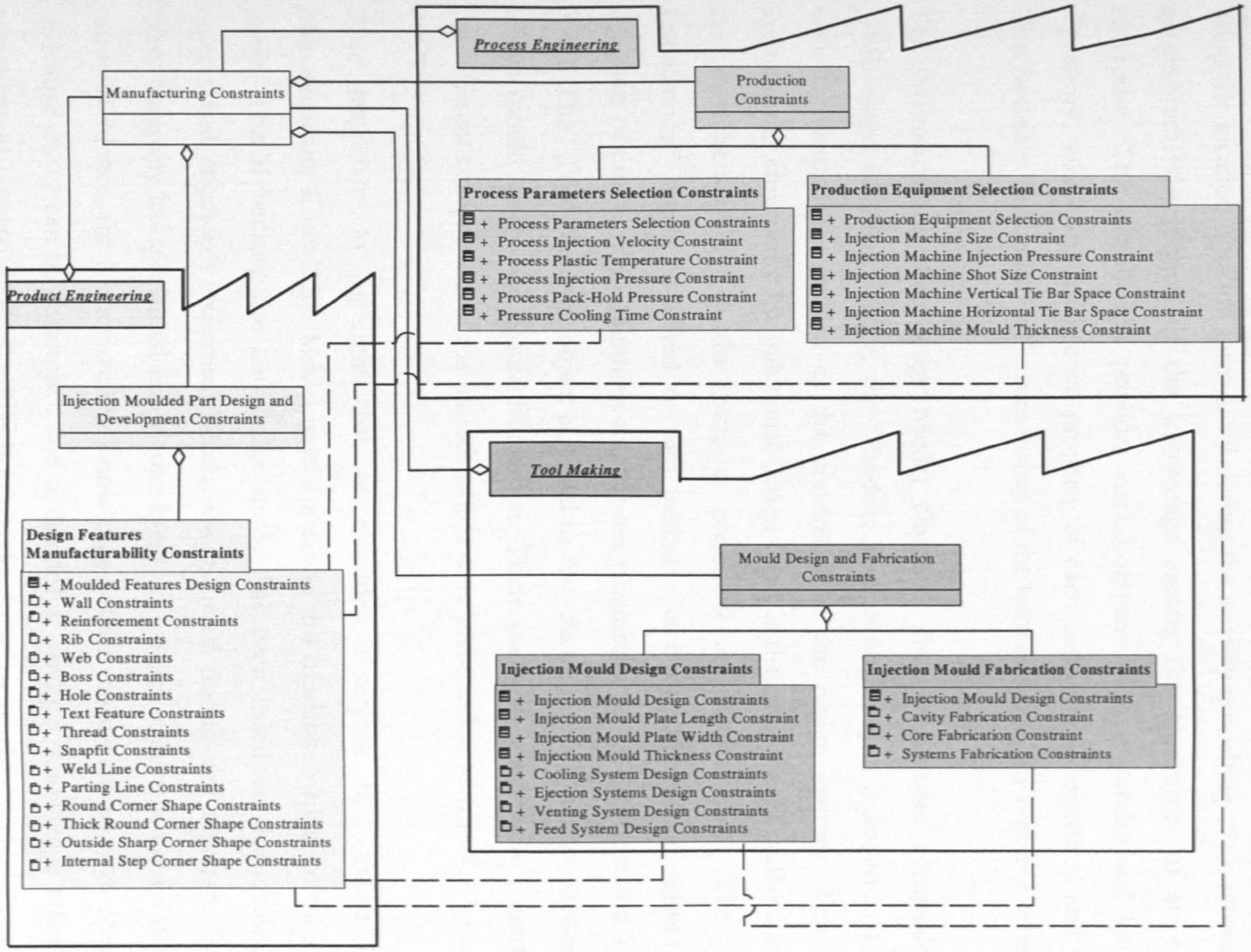


Figure 7.49 Manufacturing Knowledge Model top view representation using UML notation

In the top view of the model, the manufacturing constraints are grouped into packages according to the activity they support. These packages are further grouped in a company using the location notation introduced during the activity modelling. This represents the geographical distribution of the knowledge among the companies of an extended enterprise. This distribution provides each company with a standardised knowledge repository, where the intellectual property of each company is structured and secured. This facilitates the update and maintenance of the knowledge at any required time.

The Manufacturing Knowledge Model illustrates three instantiated companies using UML object notation. Hence, the "Product Engineering" object is linked to a package with the knowledge related to the constraints of the design features. This package includes the constraints for individual design features that must be considered to ensure the manufacturability of the part, as presented in section 7.4.1. The "Process Engineering" object is linked to two packages with the knowledge related to the selection of production equipment and process parameters presented in section 7.4.2 and 7.4.3. The "Tool Making" object is linked to two packages with the knowledge related to the mould design and mould fabrication. These packages contain the constraints for the different components of the mould, which were presented in section 7.4.4 and 7.4.5.

It is important to highlight that an extended enterprise that implements the Manufacturing Knowledge Model must customise this model according to their specific needs. This is because the knowledge model has been based on documents, books, reports and engineer's expertise, which was gathered during this research. However, every company has specialised knowledge based on specific requirements and intellectual property. Hence, the Manufacturing Knowledge Model was structured as an open model to ensure that it can be customised and updated. For example, knowledge related to the environment constraints for the injection moulded part or machining constraints for the mould design and fabrication could be added by extending the classes in the model. It can also be extended to cover knowledge related to other engineering activities if required.

The interaction between manufacturing constraints are represented in the top view of the Manufacturing Knowledge Model (see figure 7.49) using a dotted line connected among the different packages of the geographically distributed partners. The following section describes these interactions in more detail.

7.5.1 Representation of the integrity of the Manufacturing Knowledge Model

As mentioned in section 7.2.1 there are two types of interactions between the manufacturing constraints:

- Interactions within the same manufacturing constraint: these are represented in the methods of a manufacturing constraint class.
- Interactions among manufacturing constraints: these are represented using dotted arrows between classes.

Table 7.3, 7.4 and 7.5 present, in detail, the interactions among the manufacturing constraints of the Manufacturing Knowledge Model. One table is presented for each instantiated company object: product engineering, process engineering and tool making. In these tables, the different rows present the constraints according to the entity on which they are applied, for example, the second row of table 7.3 contains the manufacturing constraints applied on the wall. For identification purposes, the columns and rows are named after letters and numbers. Hence, any manufacturing constraint in the table can be identified with the pattern (letter)(number). As such, the cell A2 represents the Wall Thickness Constraint.

The manufacturing constraint(s) that interact with another constraint are represented within parenthesis in each of the cells of the table. The following paragraphs present some examples of these interactions:

- Interaction between gate position and walls, bosses as well as holes: The recommendation rule for positioning the gate highlighted an interaction among the gate position, which is represented by the "Gating System Positions Constraint" class

(Constraint D31), and the features of the part, which are represented by the "Design Features Manufacturability Constraints" class (Constraint A1). The notation used to represent this interaction in the Manufacturing Knowledge Model was a dotted arrow directed from the "Design Features Manufacturability Constraints" class to the "Gating Positions Constraint" class.

- Interaction between vents position and gating system: Another interaction is between the venting position (Constraint B20) and the gating system (Constraint A31). This is because the position of the vents depends on the position of the gate(s), and therefore the latter should be firstly placed. This is represented in the interaction between the "Gating System Design Constraints" and the "Vent Positions Constraint" class.
- Interaction between vents position and bosses, holes and ribs: In addition, the position of vents (Constraint B20) is related to possible weld lines. Therefore, it is required to consider the manufacturability of the part features that could cause a weld line, such as bosses (Constraint A5), holes (Constraint A7) and ribs (Constraint A4), before considering the position of the vents. This is formally represented with an interaction between the "Vent Positions Constraint" class and the "Boss Constraints", "Hole Constraints" and "Rib Constraints" classes.
- Interactions between boss and wall: The manufacturability of the wall (Constraint A2) should be ensured before considering the thickness of the boss (Constraint B5). This is because without ensuring that the wall is within manufacturability limitations, any further consideration is meaningless. Therefore, another interaction is represented between the "Boss Thickness Constraints" class and the "Wall Constraints" class.

Like these examples, all the interactions among the manufacturing constraints classes of the Manufacturing Knowledge Model were represented using the same approach.

Table 7.3 Manufacturing constraints located in the product engineering site

	A	B	C	D	E	F
1	Design Features Manufacturability Constraints					
2	Wall Constraints	Wall Thickness	Wall Transition	Wall Draft Angle		
3	Reinforcement Constraints	Reinforcement Draft Angle	Reinforcement Base Radius	Reinforcement Distance (←A2)		
4	Rib Constraints	Rib Width (←A2)	Rib Height (←A2)			
5	Boss Constraints	Boss Thickness (←A2)	Boss Height			
6	Web Constraints	Web Width (←A5, A2)	Web Height (←A2)	Web Length (←A2)		
7	Hole Constraints	Hole Draft Angle	Hole Depth	Hole Direction	Hole Distance	
8	Snapfit Constraints	Snapfit Radius (←A2)	Snapfit Thickness	Snapfit Entrance Side Angle	Snapfit Retaining Side Angle	Snapfit Undercut
9	Thread Constraints	Thread Draft Angle	Thread Internal Cylinder Screw Length	Internal Thread Screw Diameter	Thread Undercut	
10	Weld Line Constraints	Weld Line Positions (←A5, A7)				
11	Parting Line Constraints	Parting Line Positions (←A9)				
12	Round Corner Shape Constraints	Round CS Outside Radius (←A2,C12)	Round CS Inside Radius Constraint (←A2)			
13	Thick Round Corner Shape Constraints	Thick Round CS Outside Radius (←A2,C13)	Thick Round CS Inside Radius Constraint (←A2)			
14	Outside Sharp Corner Shape Constraints	Outside Sharp CS Inside Diameter				
15	Internal Step Corner Shape Constraints	Internal Step CS Width	Internal Step CS Height	Internal Step CS Outside Radius (←A2)		

Table 7.4 Manufacturing constraints located in the process engineering site

	A	B	C	D	E	F
16	Production Equipment Selection Constraints	Injection Machine Size (←A1)	Injection Machine Injection Pressure	Injection Machine Shot Size (←A1)	Injection Machine Mould Thickness (←A18)	Injection Machine Vertical and Horizontal Tie Bar Space (←A18)
17	Process Parameters Selection Constraints	Injection Velocity (←A5, A7)	Plastic Processing Temperature (←A5, A7)	Injection Pressure (←A5, A7)	Pack Hold Pressure	Cooling Time (←A2)

Table 7.5 Manufacturing constraints located in the tool making site

	A	B	C	D	E	F
18	Injection Mould Design Constraints	Injection Mould Plate Length (←A1)	Injection Mould Plate Width (←A1)	Injection Mould Thickness (←A1)	Standard Mould	
19	Cooling System Design Constraints	Cooling System Distance	Cooling System Diameter	Cooling System Layout	Cooling System Weld Line (←A5, A7)	Cooling System Circuit Type
20	Venting System Design Constraints (←D19)	Vent Positions (←A31, A21, A4, A5, A7, A11)	Vent Land Depth	Vent Land Length	Vent Relief Depth	Vent Width
21	Ejection System Design Constraints (←D19)	Ejection System Type (←A8, A9, A4, A5, A6)	Ejection System Designer Restriction			
22	Pin Ejection System Design Constraints	Pin Ejection Diameter (←A2)	Pin Ejection Head Thickness	Pin Ejection Head Diameter (←A2)	Pin Ejection Positions (←A2, A4, A5)	
23	Two Step Pin Ejection System Design Constraints	Two Step Pin Ejection Internal Diameter	Two Step Pin Ejection Head Thickness	Two Step Pin Ejection Positions (←A4, A5)		
24	Blade Ejection System Design Constraints	Blade Ejection Positions (←A4, A6)				
25	Sleeve Ejection System Design Constraints	Sleeve Ejection Positions (←A2)				
26	Valve Ejection System Design Constraints	Valve Ejection Positions (←A2)				
27	Air Ejection System Design Constraints	Air Ejection Positions (←A2)				
28	Feed System Design Constraints (←D19)					
29	Runner System Design Constraints	Runner System Angle	Runner System Diameter	Runner System Depth	Runner System Type (←D36)	
30	Sprue Design Constraints	Sprue Angle	Sprue Radius	Sprue Major Diameter (←A2)	Sprue Nose Diameter	
31	Gating System Design Constraints	Gating System Type (←A2)	Gating System Designer Restriction	Gating System Positions (←A1)		
32	Sprue Gate Design Constraints	Sprue Gate Diameter (←A2)	Sprue Gate Draft Angle			
33	Edge Gate Design Constraints	Edge Gate Width (←A1)	Edge Gate Depth (←A2)	Edge Gate Land Length		
34	Tab Gate Design Constraints	Tab Gate Width (←A1)	Tab Gate Length	Tab Gate Depth (←A2)		
35	Diaphragm Gate Design Constraints	Diaphragm Gate Depth (←A1)	Diaphragm Gate Overlap Length (←A1)			
36	Ring Gate Design Constraints	Ring Gate Depth (←A29, A32, A2)	Ring Gate Land Length	Ring Gate Runner Type		
37	Film Gate Design Constraints	Film Gate Depth (←A2)	Film Gate Land Length	Film Gate Width (←A1)		
38	Pin Gate Design Constraints	Pin Gate Diameter (←A1)				

Furthermore, these interactions are captured among manufacturing constraints, which represent knowledge that will be geographically distributed among different companies in an extended enterprise. For example, the constraints for the decisions that are taken by the toolmaker during the "mould design" activity are highly dependant on the decisions that are taken by the product engineer. If the latter decisions were not within limitations, problems will still arise during the later stages of product development. Capturing the interactions among manufacturing constraints in such a way, provides the Manufacturing Knowledge Model with a distinctive characteristic referred to as "knowledge integrity". The integrity provides the KdCPD system with an intelligent mechanism to ensure the consideration of the knowledge related to other engineering activities, regardless the geographical location of this knowledge.

Thus, iterations would be almost eliminated because a decision cannot be taken if previous related considerations have not been taken into account. This is illustrated in figure 7.50, where the subactivities of the "Design for Manufacturing", "Injection Mould Design" and "Selection of Process Parameters" activities are now driven not only by the knowledge which constrains the decisions made during the activity, but also by the knowledge related to other activities. For example, during the gating system design, it is necessary to consider whether the design features of the part, such as walls and bosses, are within constraints. This is represented by making "Wall Constraints" and "Boss Constraints" available during the gating system design. The same applies for the venting system design, where it is necessary to consider whether or not the ejection and gating systems have been designed within limitations and also whether or not the ribs, bosses and holes of the part are within manufacturability limitations.

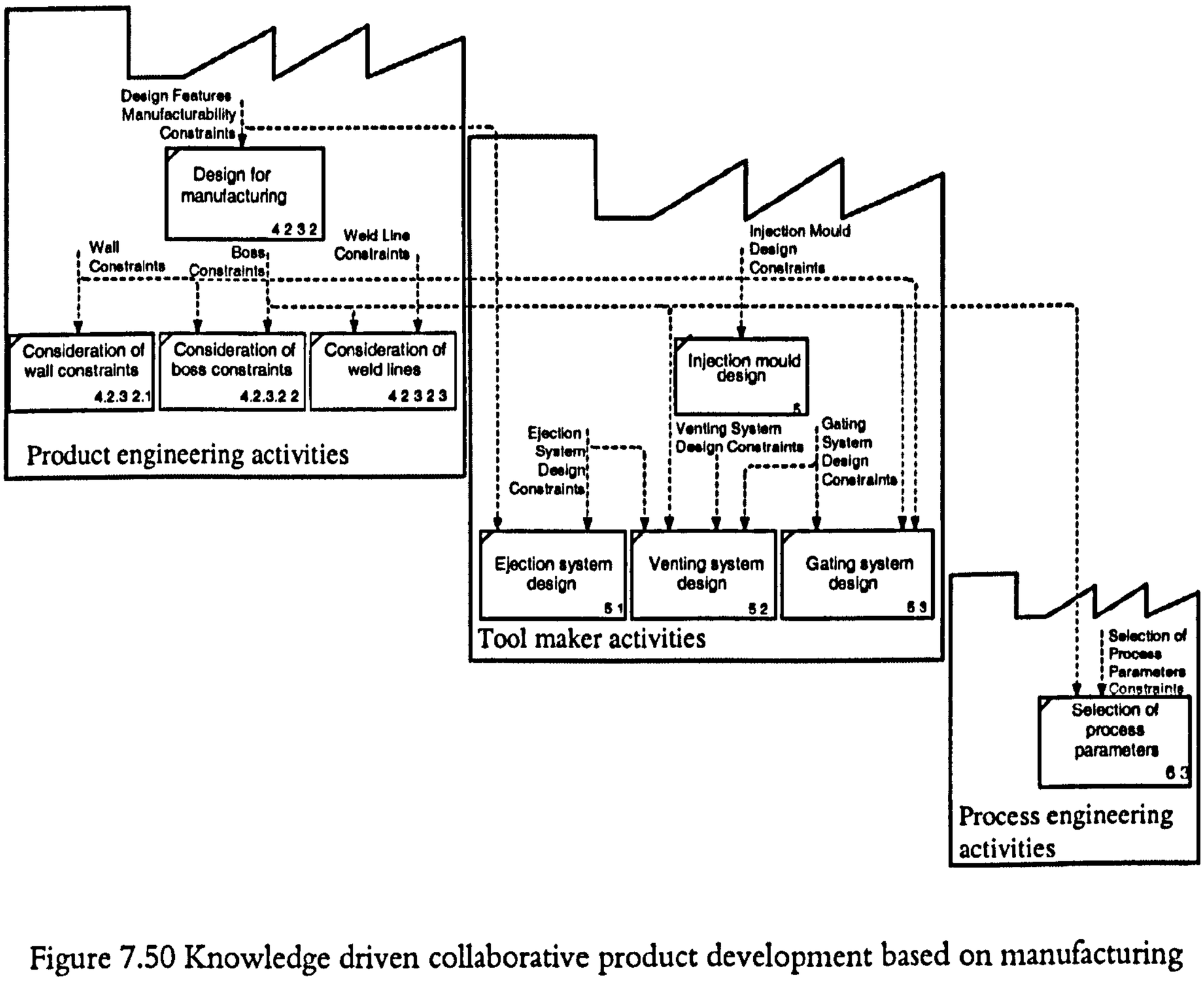


Figure 7.50 Knowledge driven collaborative product development based on manufacturing constraints

7.6 The Product Model

In addition to the Manufacturing Knowledge Model, the product data was captured in a Product Model, which captures the progress of product development. The structure of this model was determined by the fact that an injection moulded product is produced for a customer, who has a set of requirements for the part or parts that compose the product. As explained in section 7.4.1, a features based approach was used to represent the injection moulded part design during CPD.

Figure 7.51 shows the top view of the Product Model, where the product is represented as the class "Product" and has an aggregation relationship with the "Part" class. The

"Part" class has as attributes its name, description, shape description, length, width, depth, weight, production quantity and texture. It also has relationships with the classes:

- "Customer Requirements", which contains a set of requirements from the client. These requirements can be any of two types: "Design Requirement" or "Customer Requirement". Both classes inherit from the "Requirements" class. However, the "Design requirements" contains attributes for the client to specify how important the requirement is and to evaluate if this has been addressed.
- "Features", which are the different features that compose the part.
- "Plastic", which contains as attributes the characteristics of the plastic material used for the part.
- "Injection Moulding Process Information", contains information related to the process parameters for producing the part.
- "Injection Mould Information", contains the description of the mould required to produce the part.
- "Injection Moulding Machine Information", contains the description of the machine required to produce the part.

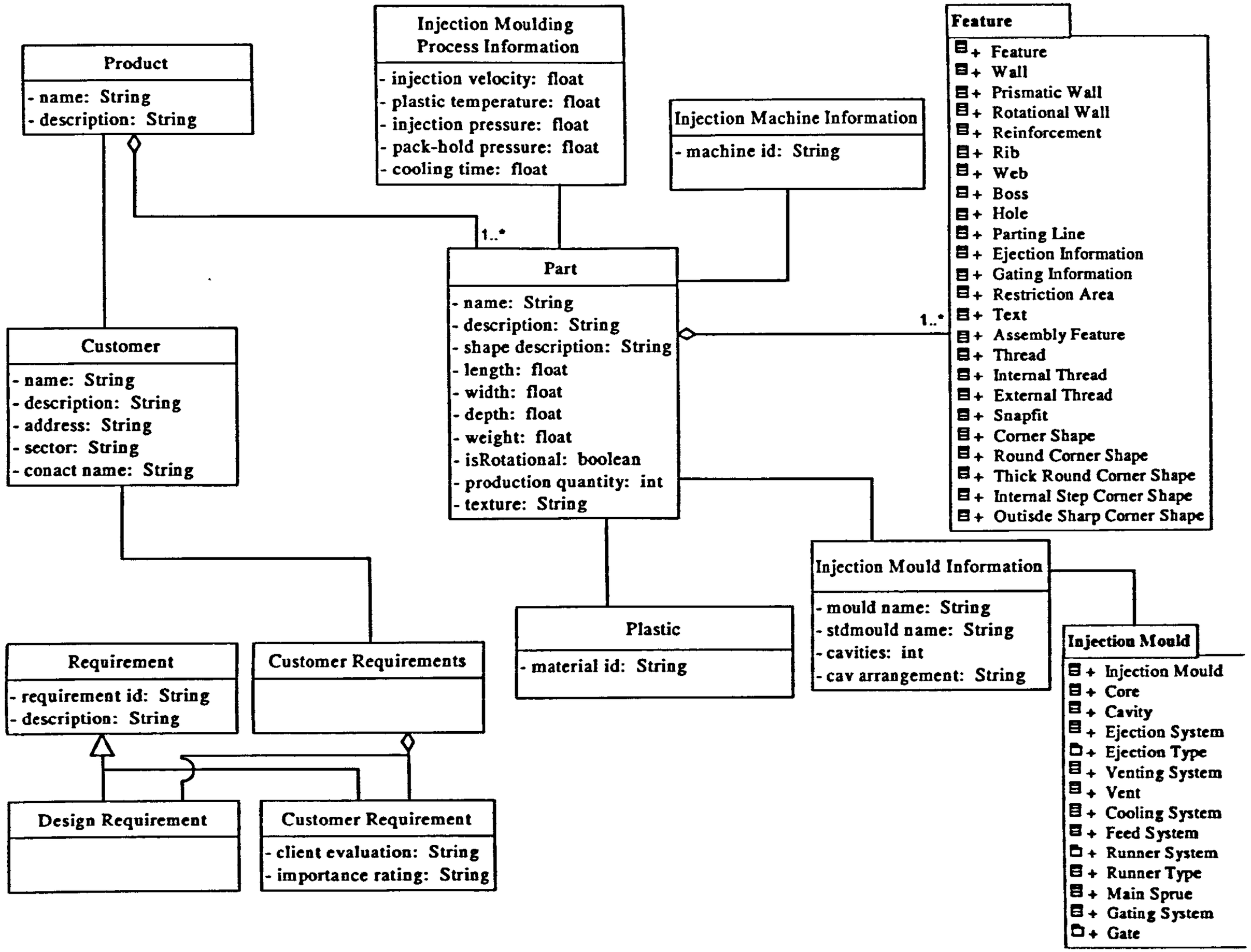


Figure 7.51 Product Model representation using UML notation

The mouldable features considered were the ones for which the manufacturing constraints were captured. These are: wall, hole, reinforcements (rib, web and boss), text, corner shape, parting line, weld line, gate position, ejection position, threads and snap fits. As figure 7.52 shows, these features are represented as subclasses of the feature class. Each feature has a name and set of attributes, which describe its dimensions and its position within the part. The feature class also contains an attribute called "isCritical" to specify if the feature is critical or not for the part's functionality. This is used to prioritise the features during the design for manufacturing analysis.

The wall feature is considered to be the main feature where other features are placed on. This relationship is represented through aggregation relationships between the "Reinforcement", "Hole", "Parting Line", "Restriction Area", "Corner Shape", "Text Feature", "Snap fit" and the "Wall" feature. In addition, the wall feature has an aggregation relationship with itself. This represents the collection of walls that are adjacent to any wall.

In addition, the data of the components of a mould is structured in the diagram shown in figure 7.53. As explained in section 7.4.4, the mould is composed of a core and a cavity. The core is composed of an ejection, venting, cooling and runner system. The cavity is composed of a venting, cooling, and feed system. The feed system is composed of a runner and gating system and a sprue. All these components are represented as classes that have an aggregation relationship with the "Injection Mould" class.

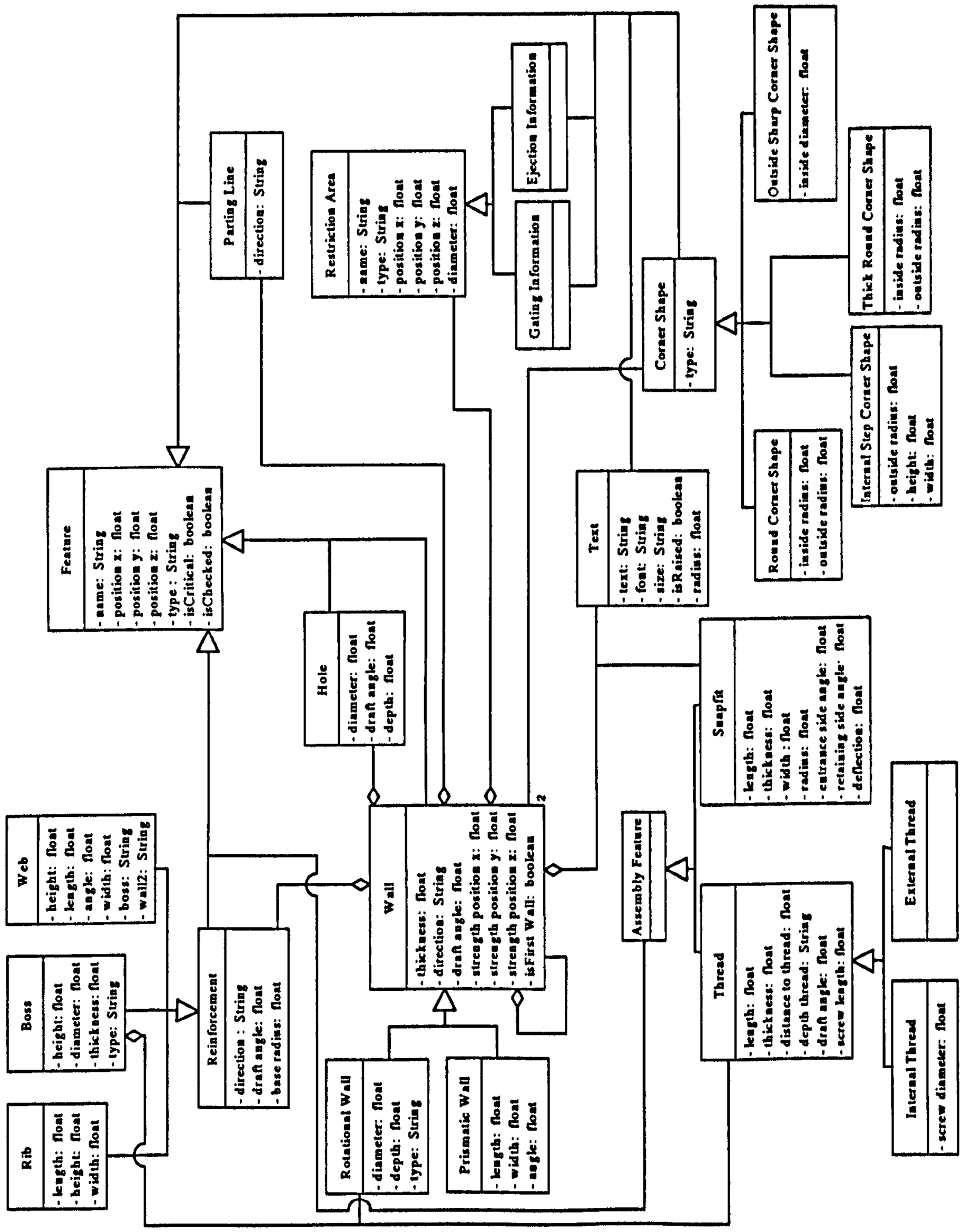


Figure 7.52 Representation of the "Features" class and its relationships using UML notation

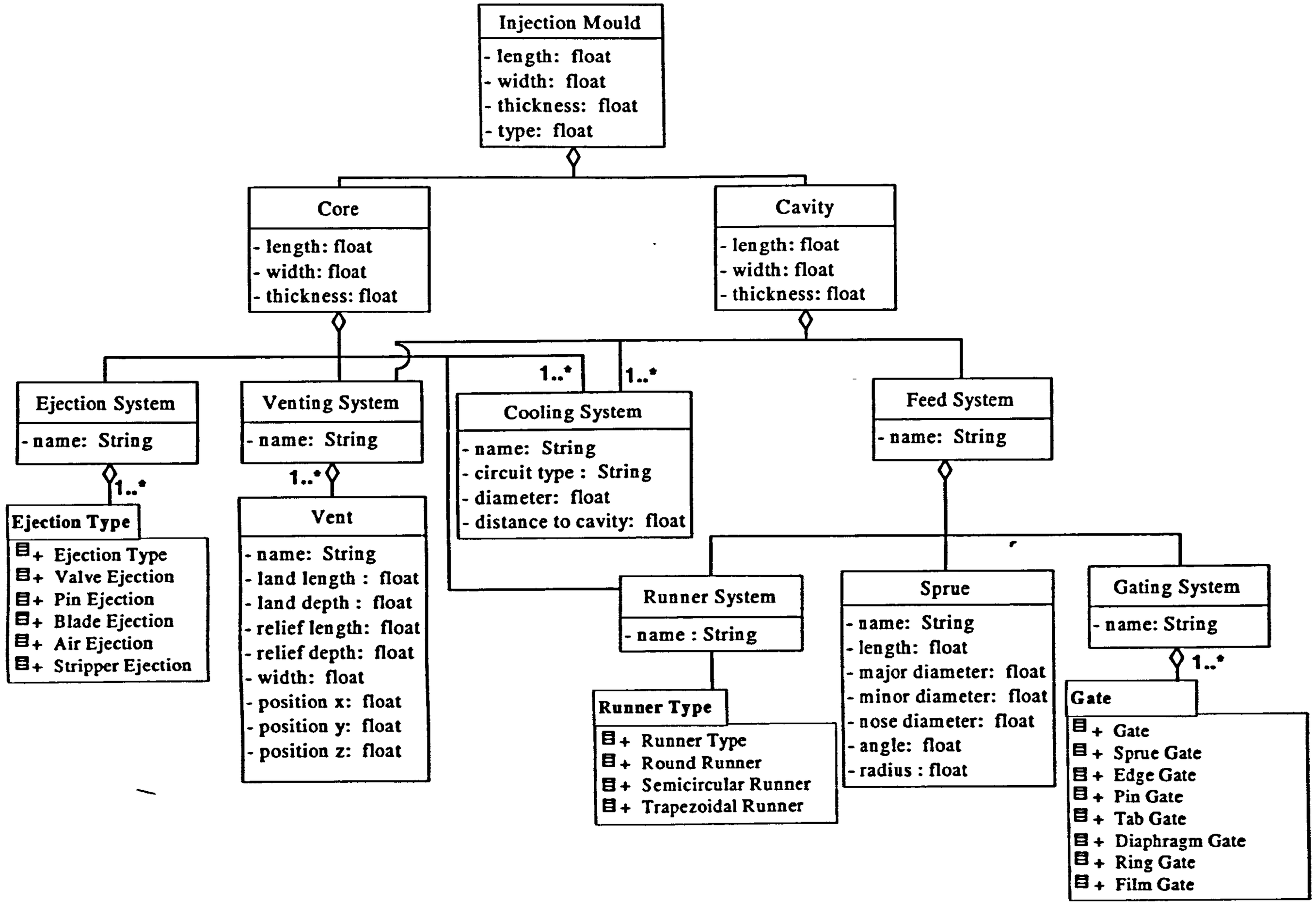


Figure 7.53 Representation of the "Injection Mould" class and its relationships using UML notation

7.7 The Organisation Model

The Organisation Model is to be the source of data related to the project and team members information. As shown in the diagram of figure 7.54, the model contains a class called "Project". This class represents that upon customer request, the company launches a project to develop the product. This project is done by a project team, which is composed of one or more employees internal and external to the company. In order to represent these relationships, the "Employee" class has a relationships with the "Project Team" and "Company" class. During the project, one of the team members is designated as the team leader to coordinate the project.

Finally, the project is composed of one or more tasks, which are represented by the "Task" class. These tasks are assigned to the employees of the geographically distributed companies. This is represented using a relationship between the "Task" class and the "Employee" class.

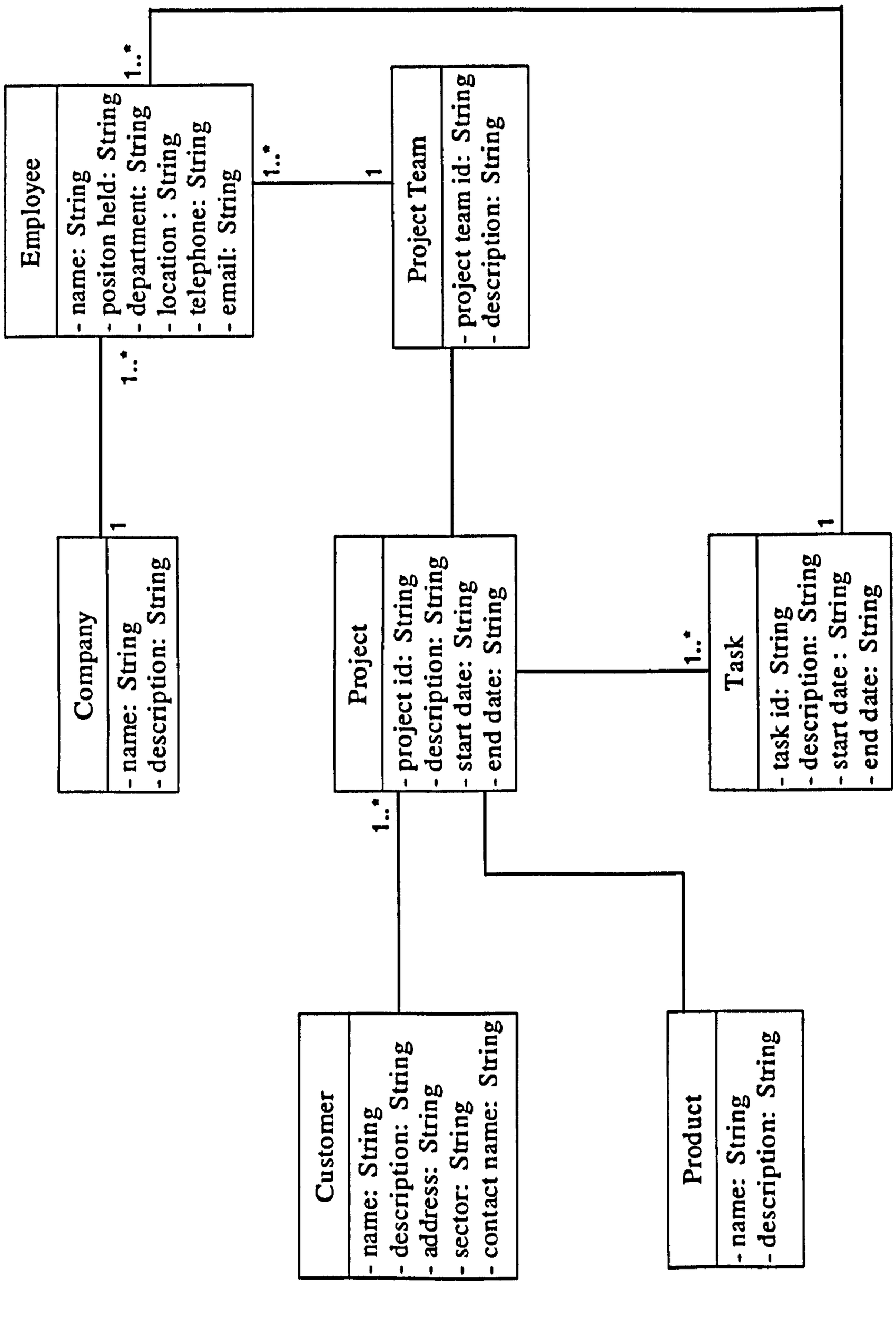


Figure 7.54 Organisation Model representation using UML notation

7.8 Closing Remarks

One of the original contributions of the research is the capture, representation, and provision of product life cycle knowledge to support engineering decision making in a collaborative environment. In order to provide this solution, the Manufacturing Knowledge Model presented in this chapter is the basis of the proposed KdCPD system architecture presented in chapter 6. In addition, the knowledge modelling process, presented in this chapter, to achieve such model is an additional methodological contribution of the research.

The Manufacturing Knowledge Model, along with the Product Model, Engineering Data Models, Product Model and Organization Model, also proposed in this chapter, will be further explored in next chapter. This is in order to demonstrate how these models can effectively support making right engineering decisions during collaborative product development.

Chapter 8

Exploring the Manufacturing Knowledge Model to Support Collaborative Injection Moulding Product Development

8.1 Introduction

This chapter presents the research performed by the author to explore, through different case studies, the provision of product life cycle knowledge and data to support engineering decision making in a collaborative environment. The case studies are scenarios of the product engineer, toolmaker and process engineer performing different engineering applications in geographically distributed locations supported by an integrated source of product life cycle data (Product Model) and knowledge (Manufacturing Knowledge Model). The engineering applications, which were identified during the activity modelling, are included in the proposed KdCPD system architecture. These are: Design for Manufacturing, Selection of Production Equipment, Selection of Process Parameters as well as Mould Design and Fabrication.

8.2 Design for Manufacturing application

As described in section 6.4.1.3, this application ensures that the functional features of the plastic part are designed within manufacturing constraints. This application is supported by:

1. The "Design Features Manufacturability Constraints" package (section 7.4.1) of the Manufacturing Knowledge Model. This package contains constraints such as "Wall Constraints", "Reinforcement Constraints", "Boss Constraints" and "Web Constraints".
2. The definition of the part's features stored in the Product Model.

3. The data of the plastic materials, which is stored in the Material Engineering Model.

During this application, the design for manufacturing (DFM) analysis is done by checking each feature's definition according to its corresponding manufacturing constraints in the Manufacturing Knowledge Model. After all the features have been checked, feedback advice is produced as a result of this analysis. This feedback contains the manufacturability problems of the part and advice of how to solve these problems. The order in which the features are checked is done according to: the features' criticality for the functionality of the part and the interactions between manufacturing constraints. In order to explore how this is performed in a collaborative environment three case studies are presented. Their objectives are the following:

1. To explore the collaboration of the product engineer with the process engineer in order to ensure the manufacturability of a prismatic part design.
2. To explore the collaboration of the product engineer with the toolmaker in order to design an injection mould only after ensuring the manufacturability of a rotational part.
3. To explore the collaboration of the product engineer with the toolmaker in order to identify the weld lines and to optimise the gate locations in a rotational plastic part.

8.2.1 Case study 1: collaborative design for manufacturing analysis of a prismatic part

This case study is using a batteries' cover plastic part, typically used in a remote control product (see figure 8.1-a). Before starting the collaboration, the product engineer defines the data of the plastic part (i.e. name, shape description, length, width, depth and plastic material) in a design session, as shown in figure 8.1-b. In addition, the data of the part's features, such as walls, ribs, and webs, is also defined. In this case study, the attributes of the "Base Wall" feature are (see figure 8.1): length of 40 mm, width of 50 mm, thickness of 6 mm, draft angle of 0° and positioned in (0,0,0) of the 3D space. The feature "Complex Boss", which is critical for the part functionality is defined as follows: diameter of 10 mm, height of 1 mm, without base radius, nor draft angle and in position (25,30,0).

The product engineer then continues defining the other features of the part using the same approach. This data represents the conceptual design and is captured in the Product Model as illustrated in figure 8.1-c.

Thereafter, the manufacturability of the part needs to be ensured between the process engineer and the product engineer, being the latter in charge of performing the analysis of the plastic part. This collaboration is done as follows:

- By sharing and providing product life cycle knowledge and data in order to ensure the manufacturability of the part.
- By providing communication tools in order to enable real time interaction when taking decisions.

These mechanisms of collaboration will be further explored in the following subsections.

8.2.1.1 Design for manufacturing analysis based on shared product life cycle knowledge and data

The analysis of the manufacturability of the plastic part can be performed by providing product life cycle knowledge stored in the Manufacturing Knowledge Model. This knowledge was represented in the design features manufacturability constraints presented in section 7.4.1. By applying these constraints to the part data stored in the Product Model, it is possible to determine whether the part feature's are within manufacturability constraints. In case they are not, it is possible to generate feedback advice in order to change the part design definition. As such, the DFM analysis can be invoked by either product or process engineer at any time, regardless the location of the engineer, of the knowledge and of the data.

The DFM analysis for the critical and non-critical features of the case study is explained in detail in the following paragraphs.

Design for manufacturing analysis of the critical features of the batteries' cover part

Usually, one or more features are critical for providing the key functionality required from a plastic part. Therefore, the Design for Manufacturing application starts by

analysing these critical features. In this case study, the DFM analysis of the critical features "Complex boss" and "Snapfit" is performed as follows:

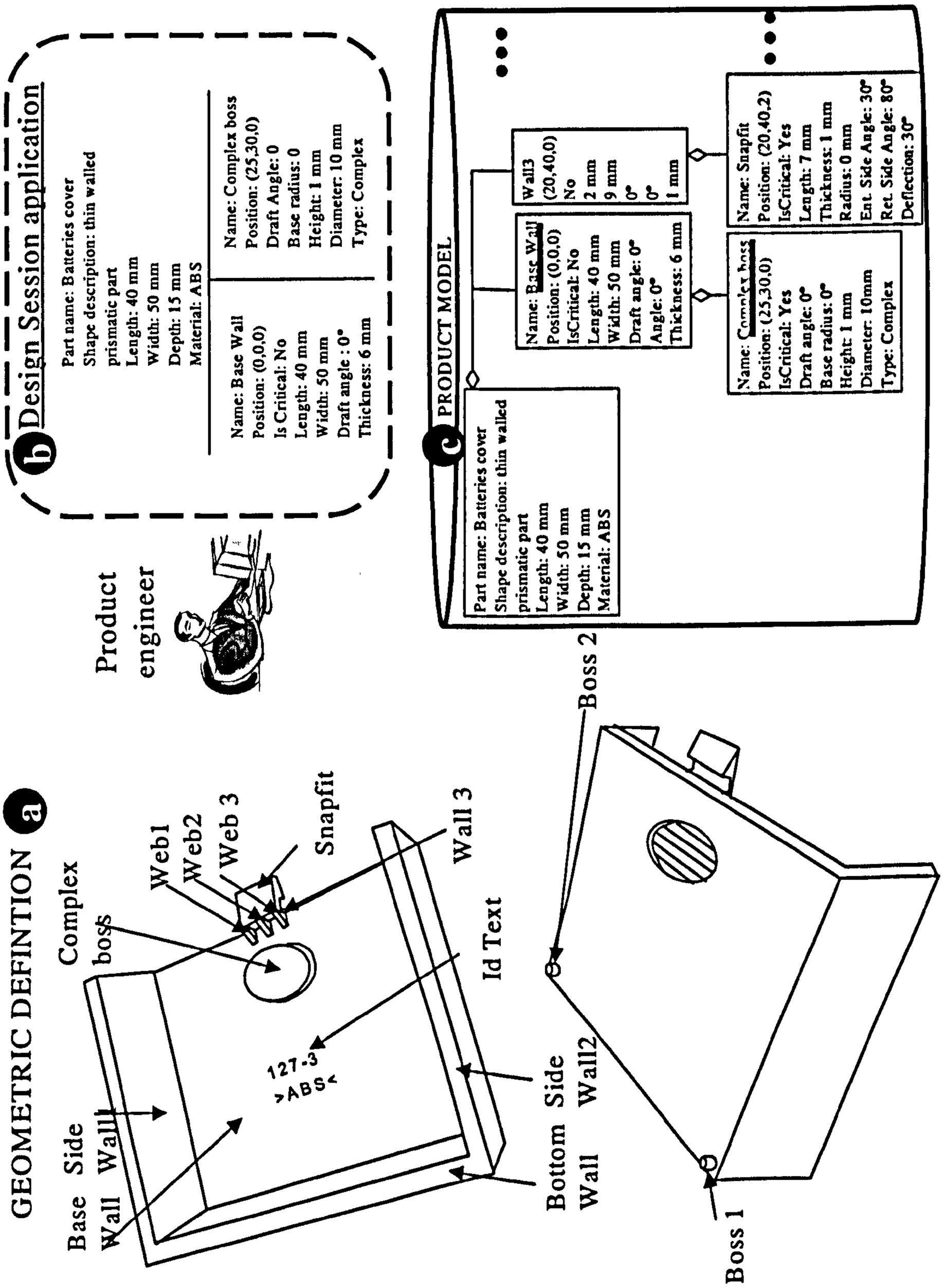


Figure 8.1 "Batteries' cover" plastic part design definition

1. **"Complex Boss" DFM analysis:** Firstly, the definition of the "Complex boss" feature is accessed by the Design for Manufacturing application from the Product Model. As shown in the Manufacturing Knowledge Model in figure 8.2, all the related constraints for the boss are invoked. These are:
    - Boss Height Constraint
    - Reinforcement Draft Angle Constraint
    - Reinforcement Base Radius Constraint

An interaction between the "Boss Height Constraint" and the "Wall Constraints" classes is detected (see section 7.5.1). As such, the manufacturability of the wall on which the boss is placed must be satisfied in order to consider the constraints imposed on the boss height. Furthermore, the boss' draft angle and base radius values are not within manufacturability constraints and hence feedback advice is produced regarding these problems. This advice includes recommended values for the draft angle and the base radius (see the Design for Manufacturing application in figure 8.2).

2. **"Snapfit" DFM analysis:** The data of the "Snapfit" feature is then accessed from the Product Model. All the constraints for the snap fit are invoked (see the Manufacturing Knowledge Model in figure 8.3):
    - Snapfit Thickness Constraint
    - Snapfit Radius Constraint
    - Snapfit Entrance Side Angle Constraint
    - Snapfit Retaining Side Angle Constraint
    - Snapfit Undercut Constraint

The "Snapfit Radius Constraint" cannot be invoked until the manufacturability of the wall on which the snap fit is placed is considered. This wall is the same as the wall on which the boss is placed. In addition, the thickness value of the snap fit is not within manufacturability constraints and feedback advice is therefore produced. The feedback advice also includes a suggestion to be aware of the side core that the mould will require

due to the undercut caused by the snap fit (see the Design for Manufacturing application in figure 8.3).

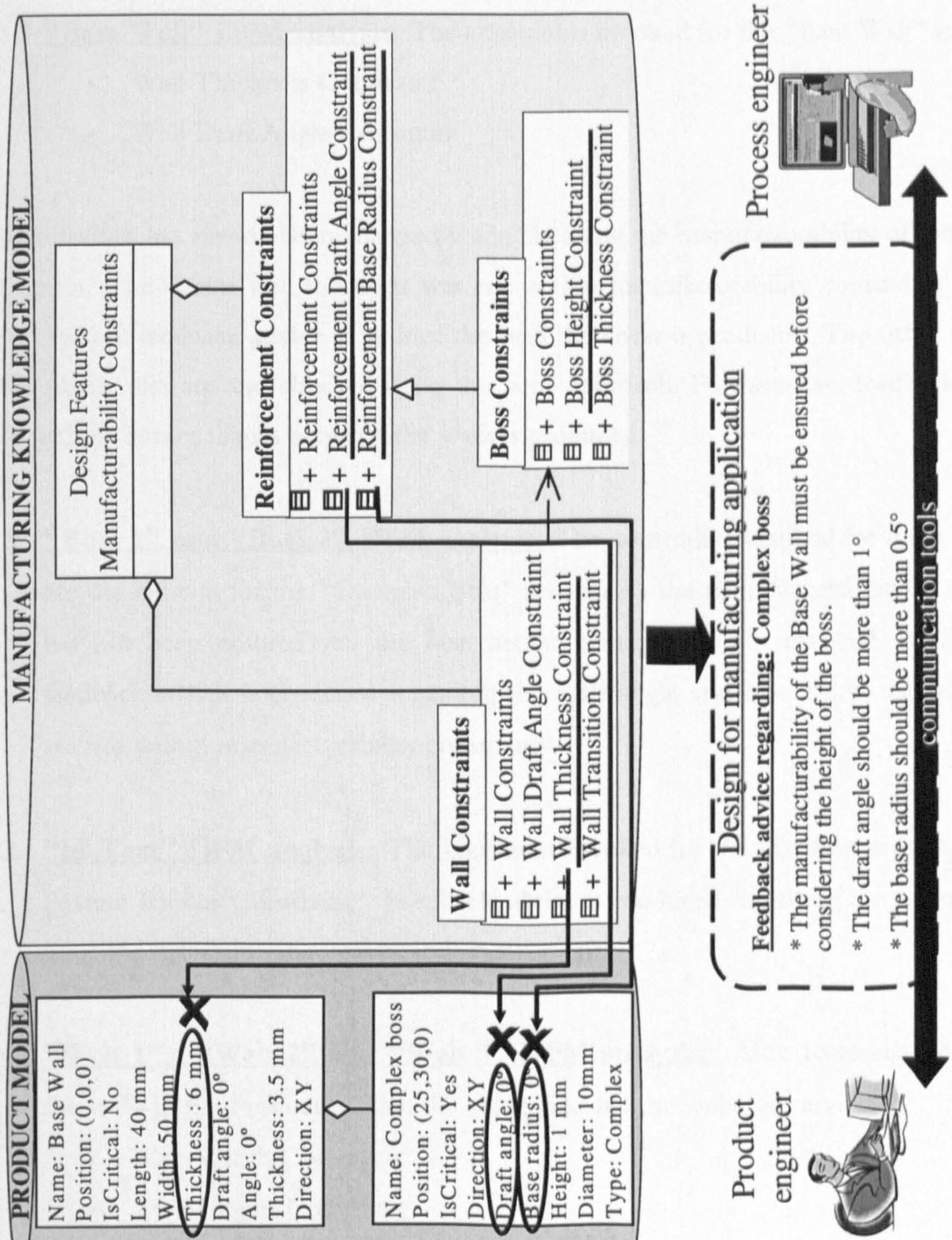


Figure 8.2 A scenario of collaborative DFM analysis of the "Complex boss" feature

Design for manufacturing analysis of the non critical features of the batteries' cover part

The Design for Manufacturing application then continues by analysing the non critical features of the part. Their DFM analysis is as follows:

1. **"Base Wall" DFM analysis:** The constraints invoked for the "Base Wall" are:
   - Wall Thickness Constraint
   - Wall Draft Angle Constraint

   This feature has already been analysed when checking the manufacturability of the critical features. The 6 mm wall thickness was not within manufacturability constraints and for this reason, feedback advice to reduce the wall thickness is produced. The other walls of the plastic part are also checked using the same approach. Furthermore, feedback advice to include corner shapes between the walls is produced.

2. **"Boss 1" and "Boss 2" DFM analysis:** The constraints invoked for these features are the same as for the "Complex Boss" feature. As the manufacturability of the wall has not been ensured yet, the boss height constraint is not invoked. In addition, feedback advice is produced regarding the draft angle and base radius values, which are not within manufacturability constraints.

3. **"Id Text" DFM analysis:** The constraint invoked for the text feature is the "Text Feature Radius Constraint". Feedback advice is produced regarding the radius value, which is not within manufacturability constraints.

4. **"Web 1" , "Web 2" and "Web 3" DFM analysis:** After accessing the data of the webs in the Product Model, the constraints for the webs are invoked:
   - Web Width Constraint
   - Web Height Constraint
   - Web Draft Angle Constraint
   - Web Base Radius Constraint

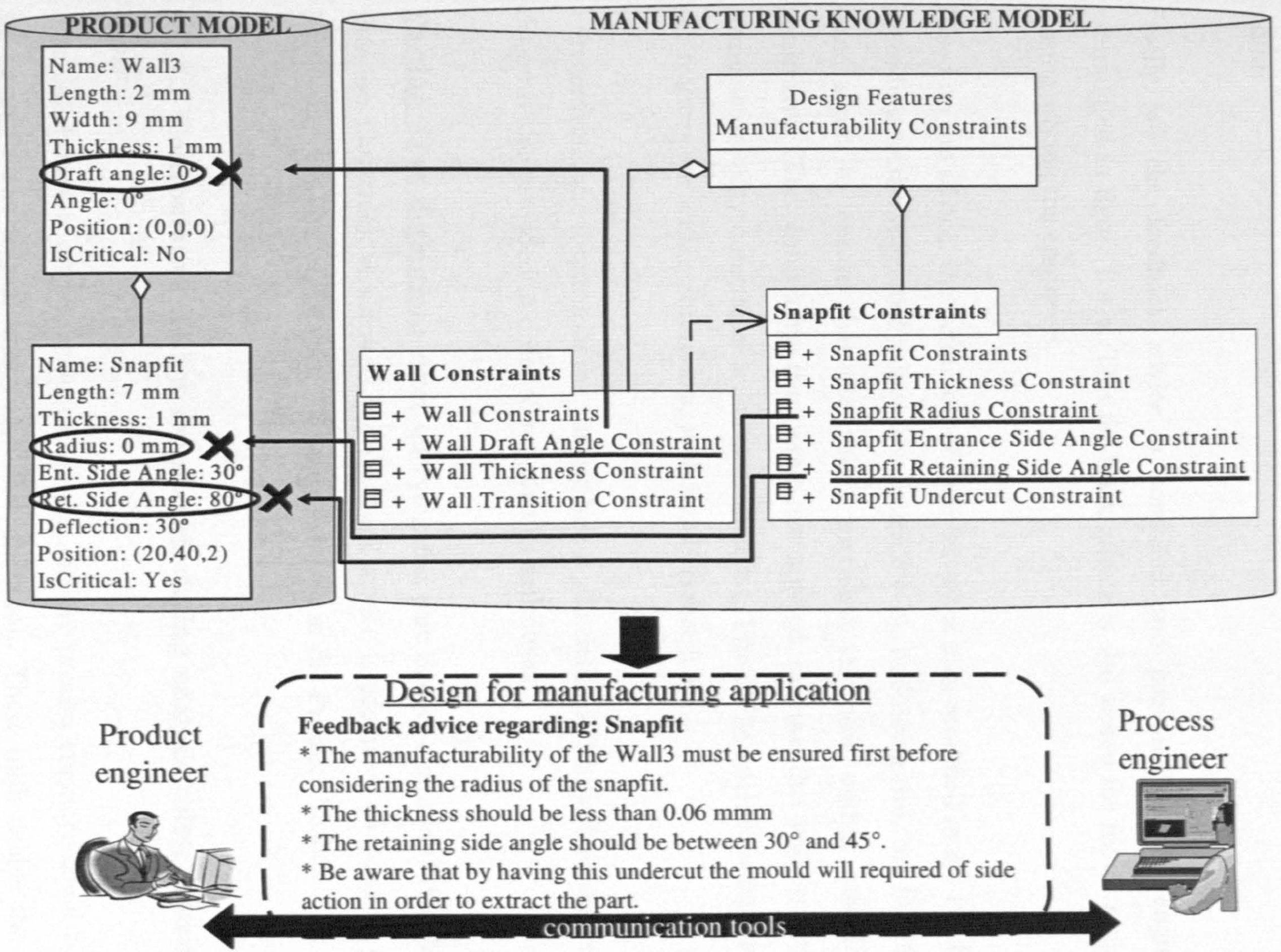


Figure 8.3 A scenario of collaborative DFM analysis of the "Snapfit" feature

The web width and height constraints cannot be invoked because the wall, on which the webs are placed, is not within manufacturability constraints. Feedback advice with this problem is produced along with other problems regarding the web draft angle and base radius.

Finally, all the feedback advice is compiled and provided to the engineer(s), as exemplified in figure 8.4-a. This feedback advice is also stored for future reviews or for sharing among the engineers.

Based on the advice, the definition of each feature is then modified in the Product Model according to the suggested values (see figure 8.4-b). Following this, the DFM analysis is done again to ensure that the values modified are now within manufacturability constraints. The constraints that were not applied because the wall was not within manufacturability constraints are now considered. The DFM analysis is repeated until all the features are within constraints, as shown in figure 8.4-c.

Furthermore, the location of the parting line is also defined. The knowledge related to its design is also provided in order to supports its definition:

Parting Line definition: After querying the part design definition, the Parting Line Position Constraint is invoked and feedback advice is produced to position the parting line at the very end of the part. This data is stored in the Product Model.

8.2.1.2 Collaborative design for manufacturing analysis using communication tools

The collaboration of the product engineer and the process engineer could be enhanced by having access to real time communication tools. These tools enable the concurrent access of the Design for Manufacturing application and real time interaction. Hence, both engineers see the feedback produced by the Design for Manufacturing application regarding the manufacturability problems of the part. This feedback and the modifications

to the part are then discussed in real time before storing the new part definition in the Product Model.

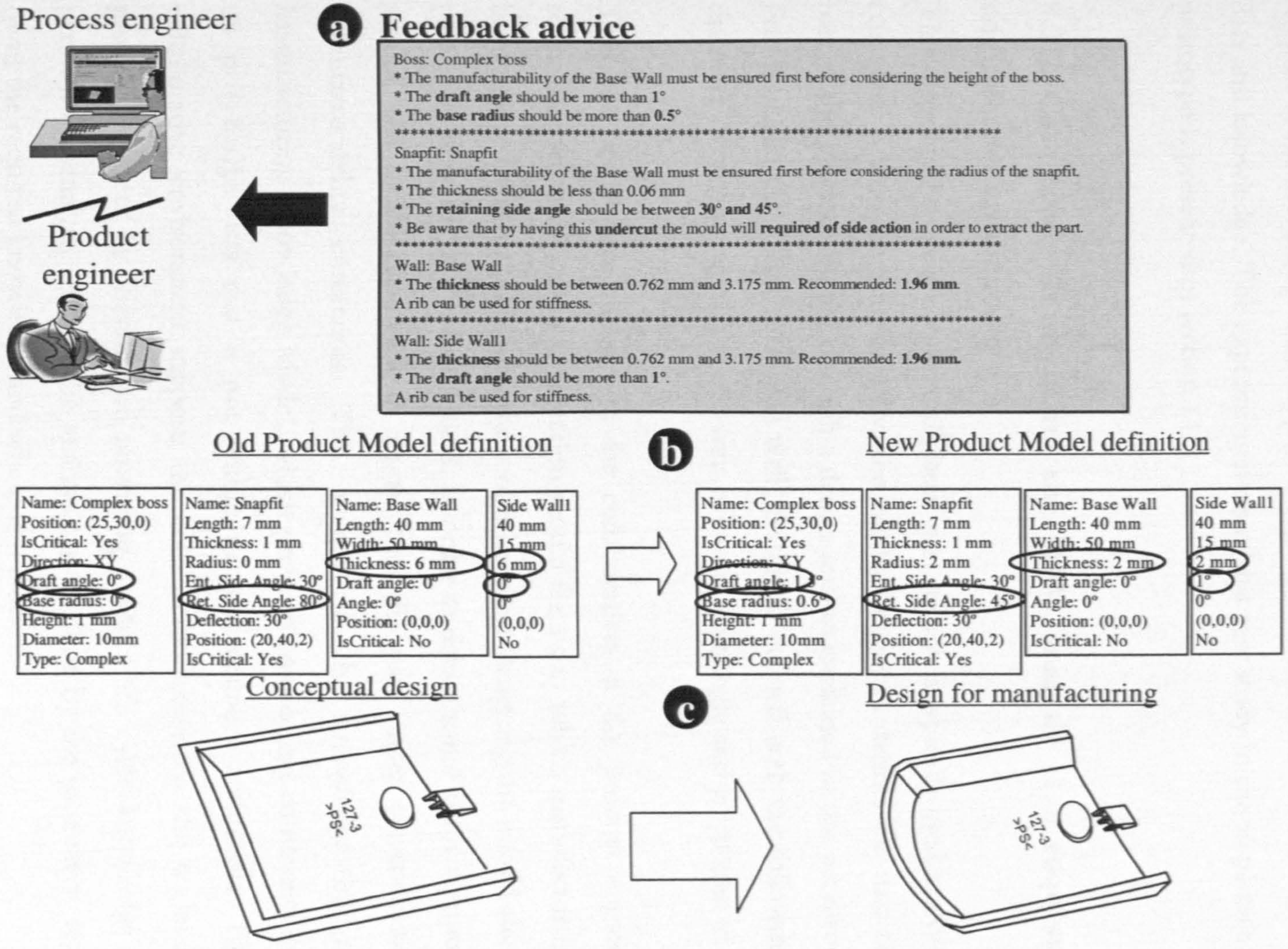


Figure 8.4 A scenario of collaborative DFM analysis among geographically distributed team members

In conclusion, this case study shows how the collaborative environment enables not only real time communication among the engineers. Even more, collaboration is taken a step further by sharing and providing product life cycle knowledge and data in real time to support decision making. This support is regardless the location of the collaborators, their data and knowledge. The experimentation of this case study in the implemented software prototype is presented in section 11.2.1.

8.2.2 Case study 2: invoking the DFM analysis as a request of other engineering application

This case study is using a rotational plastic part, which is typically used as a cap of a liquid container (see figure 8.5). Initially, the product engineer defines the data of the plastic part in the Product Model through a design session explained in the previous case study. For example, the "Disk rotational wall" feature is defined with the following attributes: diameter of 50 mm, thickness of 3 mm, without draft angle and positioned in (0,0,0).

This case study aims to explore the collaboration of the product engineer and the toolmaker when designing an injection mould for a part which manufacturability has not been ensured. In this case study, the toolmaker starts designing the mould after data of the part is available in the Product Model. In order to design some of the components of the mould, the toolmaker needs to ensure that the part under consideration is within manufacturability constraints. This is due to the knowledge integrity of the Manufacturing Knowledge Model, which prevents considering constraints that are based on part design data that is not within manufacturability constraints. The proposed collaborative environment supports the collaboration between the toolmaker and the product engineer by sharing and providing product life cycle knowledge and data to perform this analysis. Hence, this analysis can be done by the toolmaker, regardless who owns the required knowledge and where it is located.

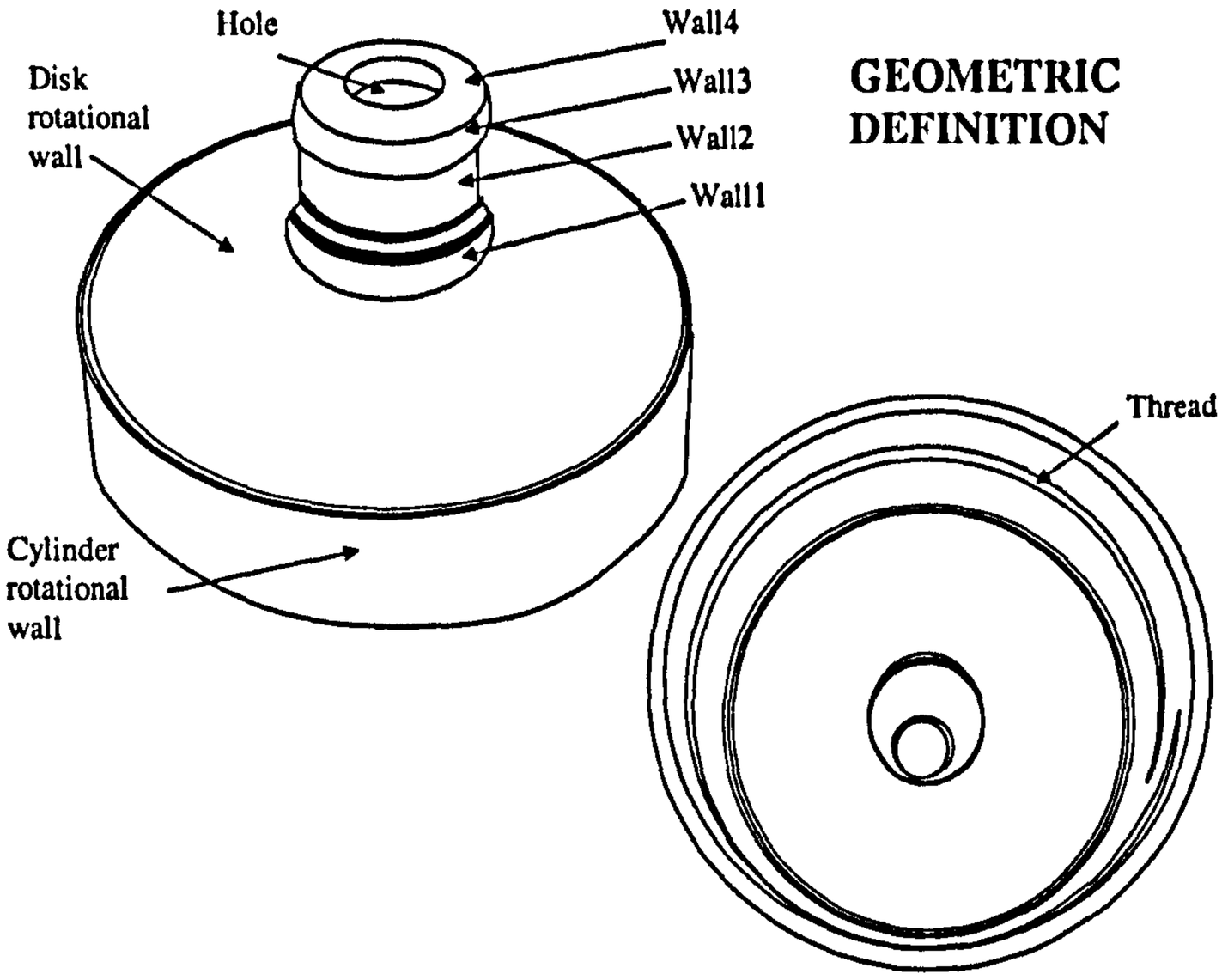


Figure 8.5 "Liquid container cap" plastic part design definition

The Mould Design and Fabrication application supports the toolmaker to perform DFM analysis, before supporting decisions regarding the mould design, by accessing the shared product life cycle knowledge and the data features' definition. The features are analysed as follows:

- **"Wall1", "Wall2", "Wall3" and "Wall4" DFM analysis:** these features are critical for the part's functionality. As such, their constraints are firstly invoked and it is determined that the draft angle of the wall is outside manufacturability constraints. Hence, feedback advice is produced regarding the need to ensure the manufacturability of the walls before designing the mould (see the Mould Design application in figure 8.6).

- **"Thread" DFM analysis:** The constraints for the thread feature are then invoked and it is determined that the draft angle and the screw length are not within constraints. Feedback regarding this problem is produced.

- **"Disk Wall" and "Cylinder Wall" DFM analysis:** Both of these are within manufacturability constraints so there is no feedback produced.

The feedback advice produced is delivered to the toolmaker, who then collaborates with the product engineer in order to ensure the manufacturability of the part design. This collaboration is supported again by sharing and providing product life cycle knowledge and data and by the provision of communication tools. As such, the toolmaker and product engineer can collaborate in either of the following ways:

1. By the toolmaker accessing the Design for Manufacturing application. This application produces feedback advice, based on product life cycle knowledge, regarding the modifications that need to be done to ensure the manufacturability of the part. The toolmaker then follows the advice and modifies the part definition in the product Model.
2. By both toolmaker and product engineer accessing concurrently the Design for Manufacturing application supported by communication tools to interact in real time. Then both can see the feedback advice and interact to modify the part definition concurrently.
3. By the toolmaker using a communication tool to inform the product engineer of the need to ensure the manufacturability of the part. Thereafter, the product engineer accesses the Design for Manufacturing application to modify the part definition based on the feedback advice. Once the part is within manufacturability constraints, the product engineer informs this to the toolmaker using a communication tool.

At the end, the part design definition is within manufacturing constraints and it is stored in the Product Model. Thereafter, the toolmaker starts the design of the mould with the assurance that no further problems will occur due to not taking into account the manufacturability of the part design. The experimentation of this case study in the implemented software prototype is presented in section 11.2.2.

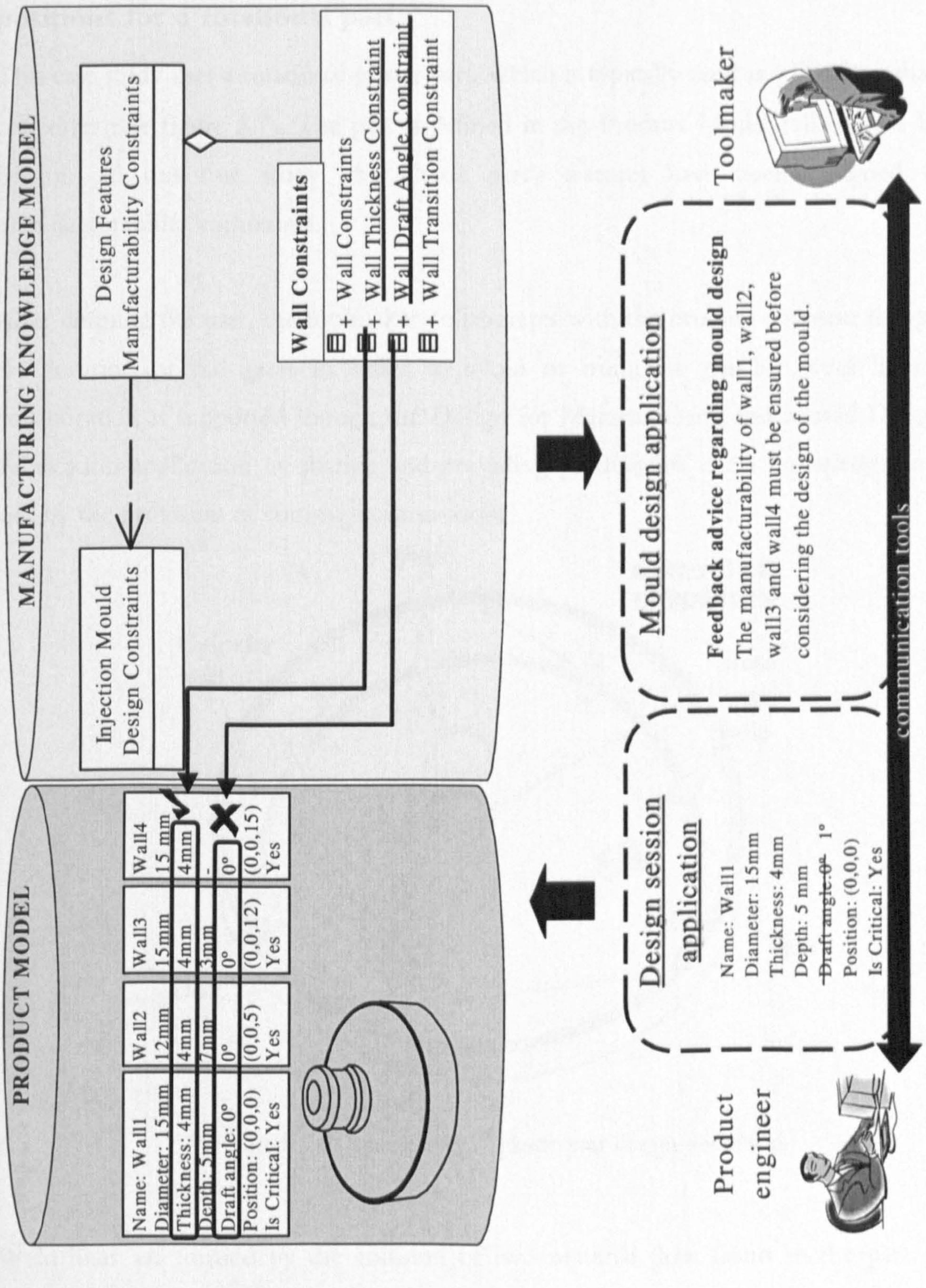


Figure 8.6 A scenario of collaborative DFM analysis of the wall feature

8.2.3 Case study 3: collaborative identification of weld lines and gate positions for a rotational part

This case study uses a rotational plastic part, which is typically used as a cap for a shampoo container (see figure 8.7). The part is defined in the Product Model following a Design Session. In this case study, the plastic part's features have been designed within manufacturability constraints.

After defining the part, the toolmaker collaborates with the product engineer to optimise the location of the gates in order to avoid or minimise possible weld lines. This collaboration is supported though the Design for Manufacturing and Mould Design and Fabrication application by sharing and providing product life cycle knowledge and data and by the provision of communication tools.

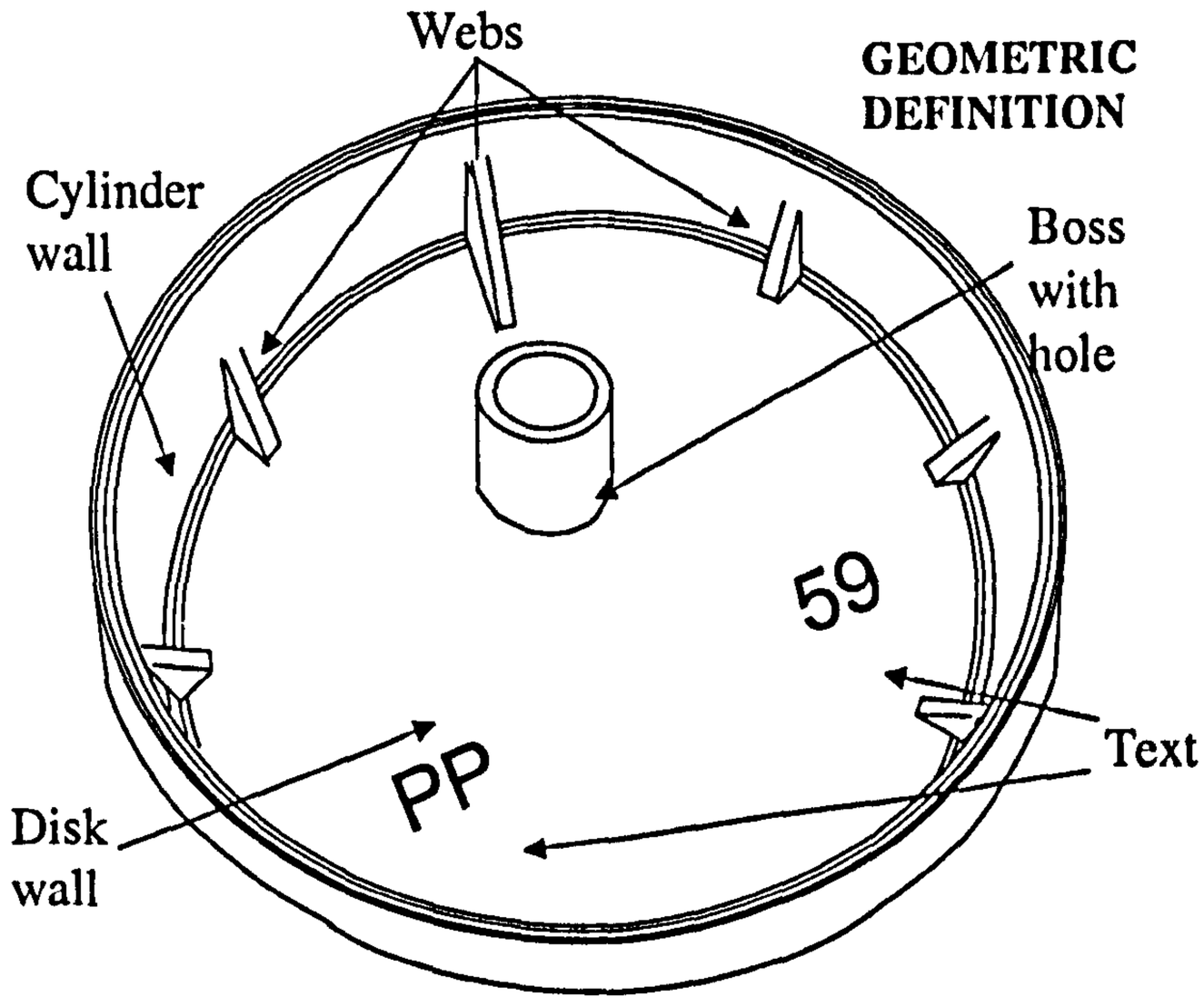


Figure 8.7 "Container cap" plastic part design definition

Weld lines are formed by the collision of two material flow fronts in the part due to obstructions, such as holes and bosses. For the identification of weld lines in the container cap plastic part, product life cycle knowledge is accessed by the applications from the Manufacturing Knowledge Model. In such a way, the product engineer requests this knowledge after ensuring the part has been designed within manufacturing constraints in

the Design for Manufacturing application. By doing this, the "Weld Line Positions Constraint", explained in section 7.4.1.7, is invoked. The constraint analyses the part geometry and detects a weld line caused by the "Boss" feature. According to this, the application produces feedback advice regarding this possible weld line and how it can be reduced by optimising the gate position.

The positions of the gates have an effect on the part appearance and therefore the selection of the suitable positions has to be done in collaboration between product engineer and toolmaker. In order to support the collaboration, the knowledge regarding the suitable gate positions is provided along with communication tools. As such, the product engineer and toolmaker can collaborate in either of the following ways:

1. By the product engineer requesting the required knowledge during the Design for Manufacturing application in order to specify preferred gate positions or in the opposite case to specify in which area of the part a gating mark is unwanted.
2. By the product engineer and toolmaker using communication tools to access the Design for Manufacturing application concurrently in order to access the required knowledge and decide which are the suitable gate positions.

In either way, the knowledge regarding the suitable gate positions is invoked by the application as follows:

- "Gating Position Constraint": the constraint analyses the geometry and determines that the centre of "Disk Wall" is a candidate position for the gate as well as the farthest point from the boss in "Cylinder Wall". This analysis is transparently performed, despite the knowledge residing in the tool making company site.

As illustrated in the Design for Manufacturing application in figure 8.8, feedback with these suggestions is provided to the product engineer and toolmaker, who collaboratively decide to place the gate in the position (0,17.5,6) of "Cylinder Wall", as it minimises the weld line and it is not on a visible surface.

This candidate gate position is stored in the Product Model and is later retrieved when designing the gating system in the Mould Design and Fabrication application (see Mould Design application in figure 8.8). The experimentation of this case study in the implemented software prototype is presented in section 11.2.3.

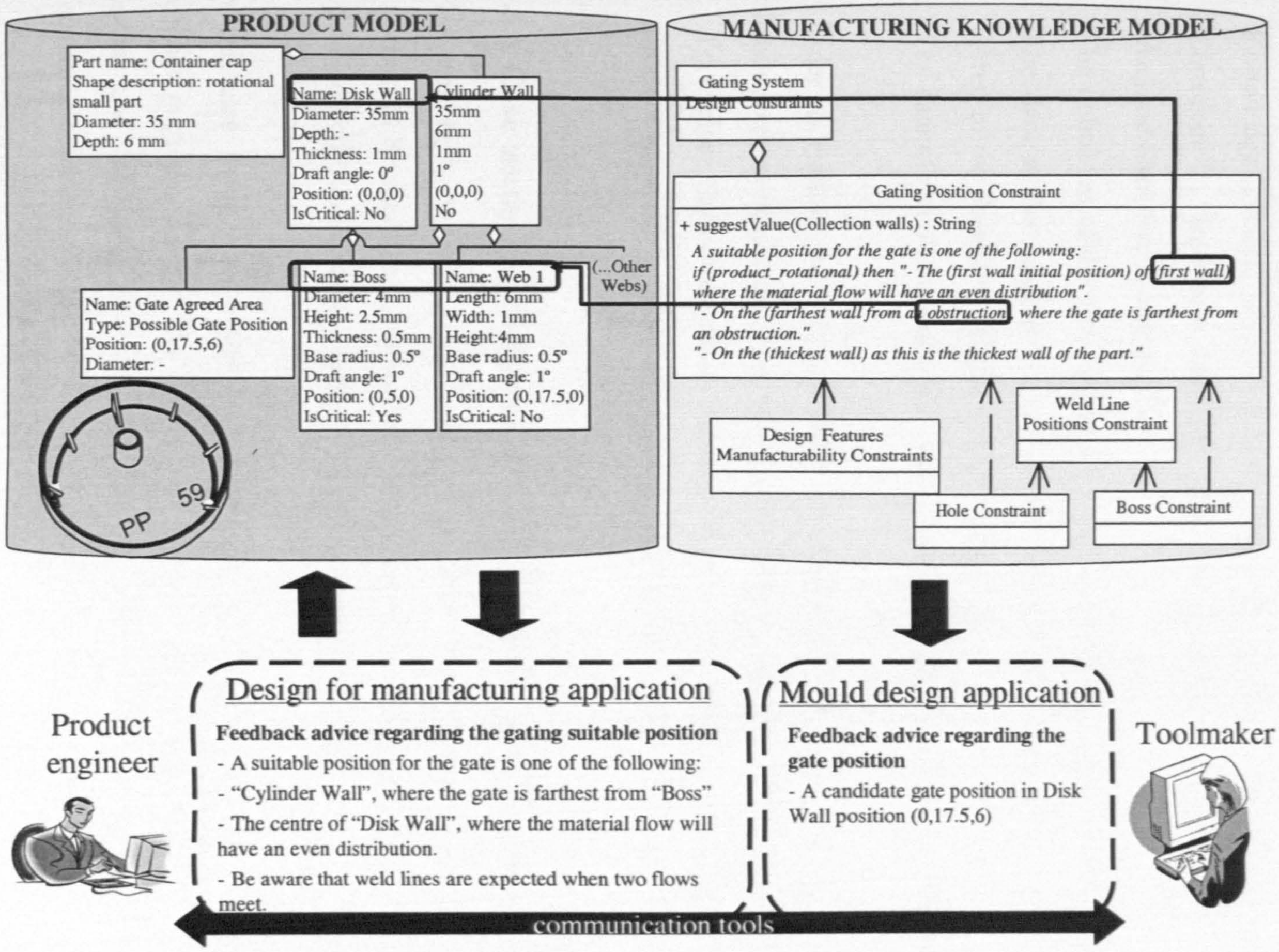


Figure 8.8 A scenario of collaborative selection of gate positions

8.3 Selection of Production Equipment application

As described in section 6.4.1.4, this application supports the selection of a suitable injection moulding machine for the production of a plastic part. This application is supported by:

- The "Production Equipment Selection Constraints" package of the Manufacturing Knowledge Model, which contains constraints such as "Injection Machine Size Constraint", "Injection Machine Injection Pressure Constraint" and "Injection Machine Shot Size Constraint".
- The definition of the part's features and the mould stored in the Product Model.
- The data of the injection moulding machines that belong to the process engineer, stored in the Resources Engineering Model.
- The data of the plastic materials, which is stored in the Material Engineering Model.

In order to explore the way in which this application supports the collaboration in a collaborative environment, a case study with the following objective is presented:

1. To explore the collaboration between the process engineer, the toolmaker, and the product engineer in order to select the most suitable production equipment.

8.3.1 Case study 4: collaborative selection of production equipment

This case study is using a connector's cap plastic part illustrated in figure 8.9. The Selection of Production Equipment application could be started without the need to have a detailed definition of all the part's features in the Product Model.

In this case study, the collaboration between the process engineer, toolmaker, and product engineer during this activity is explored. Such collaboration is achieved by sharing and providing product life cycle knowledge and data of the geographically distributed collaborators and by providing communication tools.

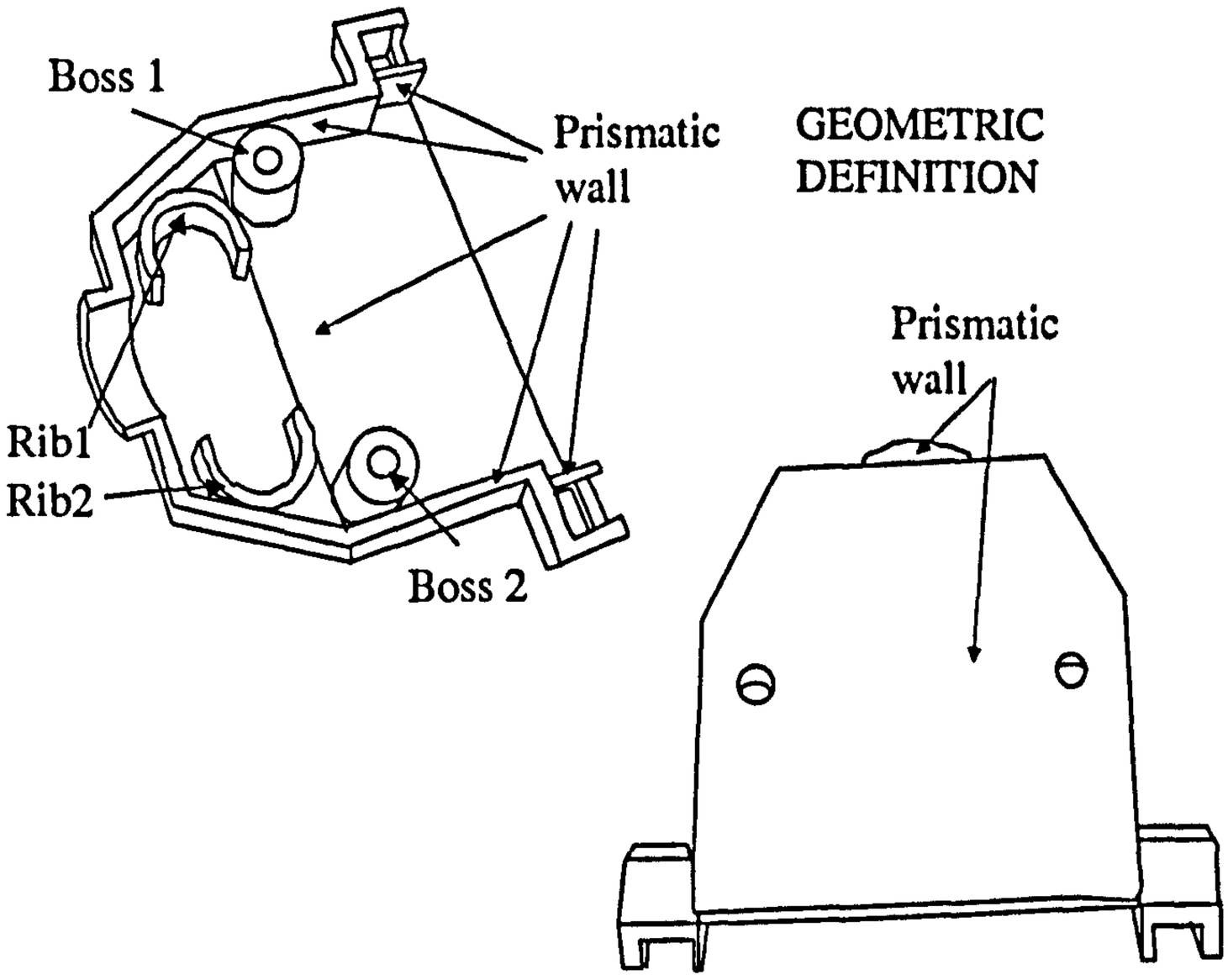


Figure 8.9 "Connector's cap" plastic part design definition

The use of these mechanisms of collaboration is explored in the following subsections.

8.3.1.1 Collaborative calculation of the suitable machine characteristics

The machine characteristics can be calculated by sharing product life cycle knowledge stored in the Manufacturing Knowledge Model. This knowledge was represented in the production equipment selection constraints presented in section 7.4.2. The machine characteristics considered are: machine size, injection pressure, shot size, minimum mould thickness and tie bar space. In addition, the number of cavities and their layout is collaboratively decided by the process engineer, toolmaker, and product engineer. This can be achieved in any of the following ways:

1. By all the collaborators communicating in real time to decide on the number of cavities and its layout.
2. By the process engineer deciding on the number of cavities and its layout. This data is then stored in the Product Model, so it is available to the toolmaker and product engineer for approval.

In this case study, 4 cavities in a balanced arrangement type 1 are selected (see figure 8.10).

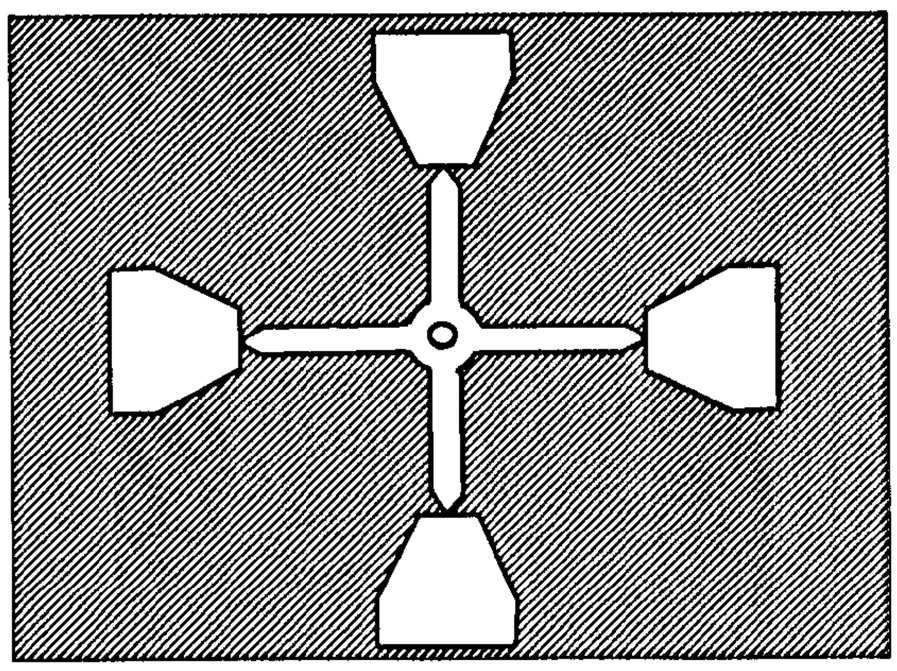


Figure 8.10 Balanced cavity arrangement for injection mould of case study 4

The constraints illustrated in the Manufacturing Knowledge Model of figure 8.11 are invoked to calculate the machine characteristics requirements. These are: machine size of 84.39 ton, an injection pressure of 1,250,000 Pa and a machine shot size of 25,234 mm^3. Furthermore, it is assumed that the mould has not yet been designed and therefore, the knowledge related to the calculation of the plate's width and length is accessed from the Manufacturing Knowledge Model in order to calculate the space in the machine that is needed to fit the mould. This knowledge is accessed transparently regardless its location, which is in the toolmaker's site. As such, the calculated tie bar space is 140 mm vertical and 200 mm horizontal.

8.3.1.2 Selection of a suitable injection machine

After the machine characteristics have been calculated, these are matched against a list of resources which belong to the process engineer. The suitable injection machine is then suggested. As illustrated in the Selection of Production Equipment application in figure 8.12, the selected injection machine is VISTA165, which has a machine size of 165 ton, a maximum injection pressure of 114,700,000 Pa, a maximum shot size of 2,540,000 mm^3, a horizontal and vertical tie bar space of 425 mm.

Based on this feedback advice, the process engineer decides to use this machine for production and stores its data in the Product Model. This new data has an impact on the

design of the mould plate, because the mould plates are required to be designed within the size limitations of the injection machine selected. As such, when the toolmaker starts the Mould Design and Fabrication application, the dimensions of the machine are automatically accessed from the Product Model, and mould plates dimensions that fit the injection machine are suggested to the toolmaker.

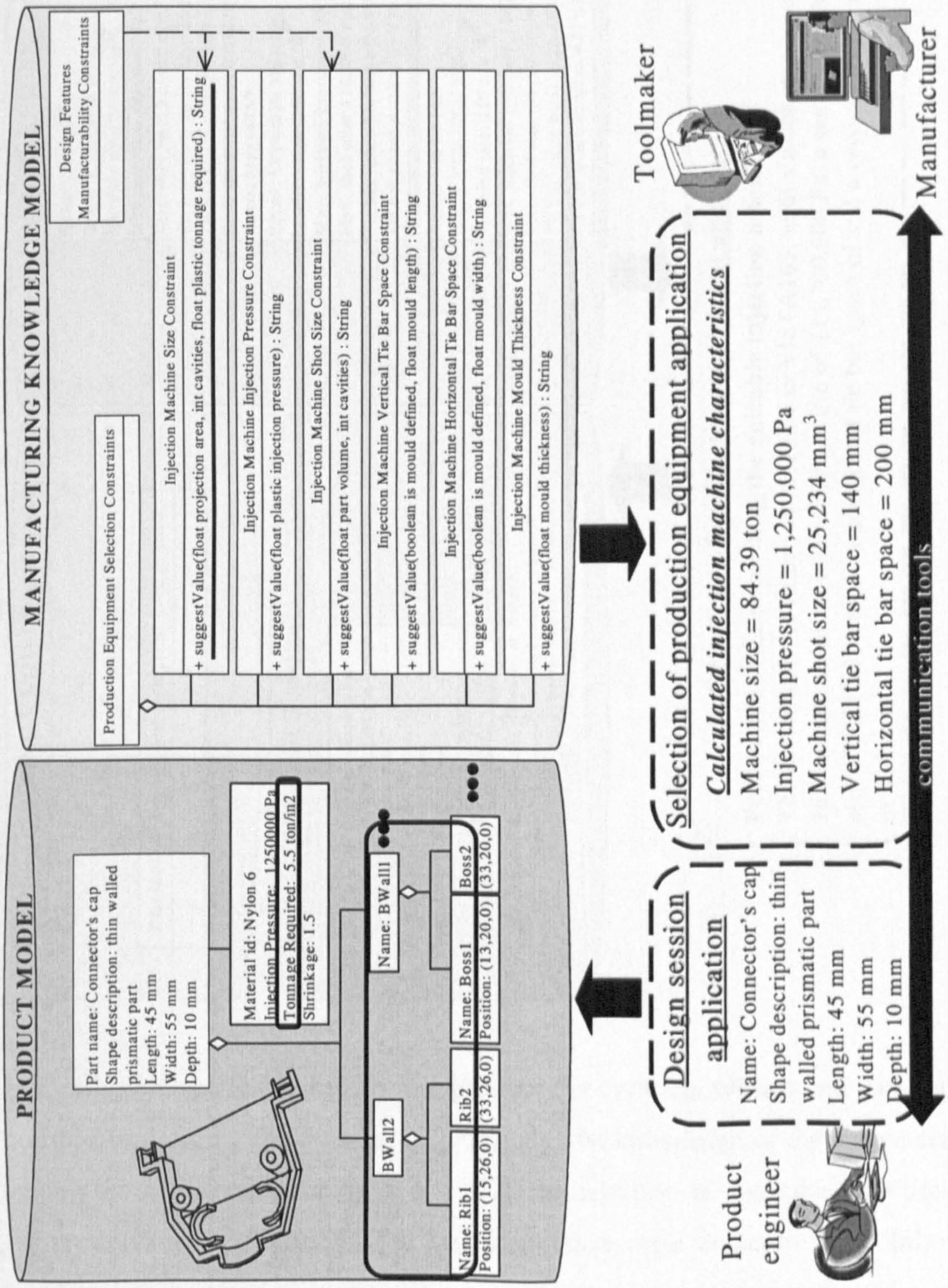


Figure 8.11 A scenario of collaborative calculation of the required machine characteristics

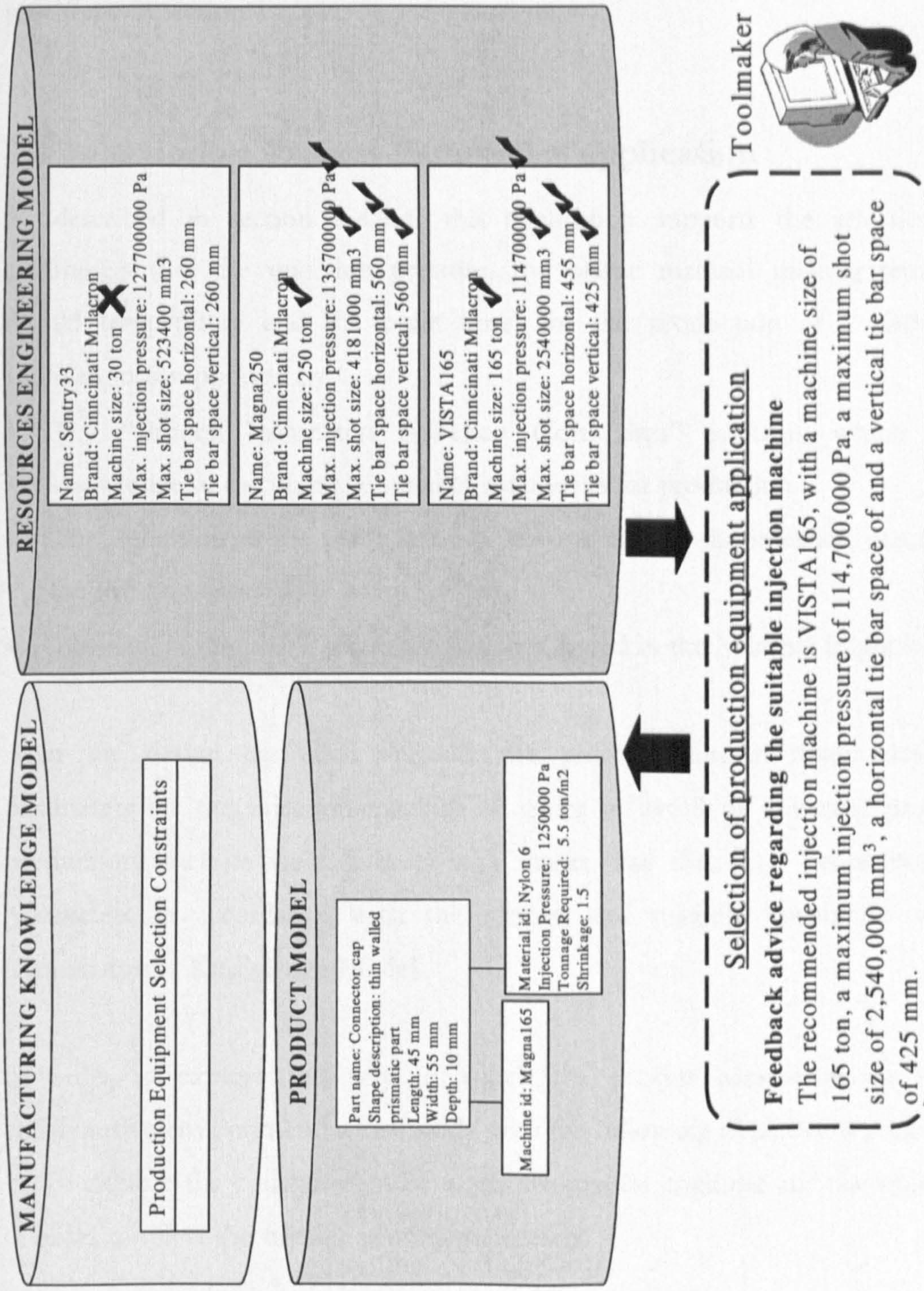


Figure 8.12 A scenario of collaborative selection of suitable injection machine

In conclusion, sharing and providing product life cycle knowledge and data supports the collaboration during this activity. For example, the knowledge of the toolmaker is shared among the collaborators in order to enable the selection of a suitable injection machine for production. Furthermore, the communication tools enhance the collaboration by enabling communication in real time in order to take decision, such as the number of

cavities or their layout. This case study is used for experimentation in the implemented prototype in section 11.3.1.

8.4 Selection of Process Parameters application

As described in section 6.4.1.4, this application supports the selection of process parameters (i.e. the injection pressure, the plastic material melting temperature, the mould temperature and the cycle time) for the production of a plastic part. This application is supported by:

- The "Process Parameters Selection Constraints" package, which contains the knowledge to calculate the suitable parameters for production.
- The definition of the part's features, the mould and the injection machine stored in the Product Model.
- The data of the plastic materials, which is stored in the Material Engineering Model.

After the design has been released, the process engineer determines the process parameters for the injection machine in order to avoid or minimise problems during production, such as weld lines or sink marks. For this, it is necessary to select the parameters in accordance with the process and material constraints stored in the Manufacturing Knowledge Model.

In order to explore the way in which the process parameters are selected in a collaborative environment, a case study with the following objective is presented:

1. To explore the collaboration between the process engineer and the product engineer in order to select the suitable process parameters.

8.4.1 Case study 5: collaborative selection of process parameters

This case study is using a printer component plastic part, which is a long thin walled prismatic part (see figure 8.13). In the proposed collaborative environment, the collaboration between the process engineer and the product engineer is achieved by sharing and providing product life cycle knowledge and data.

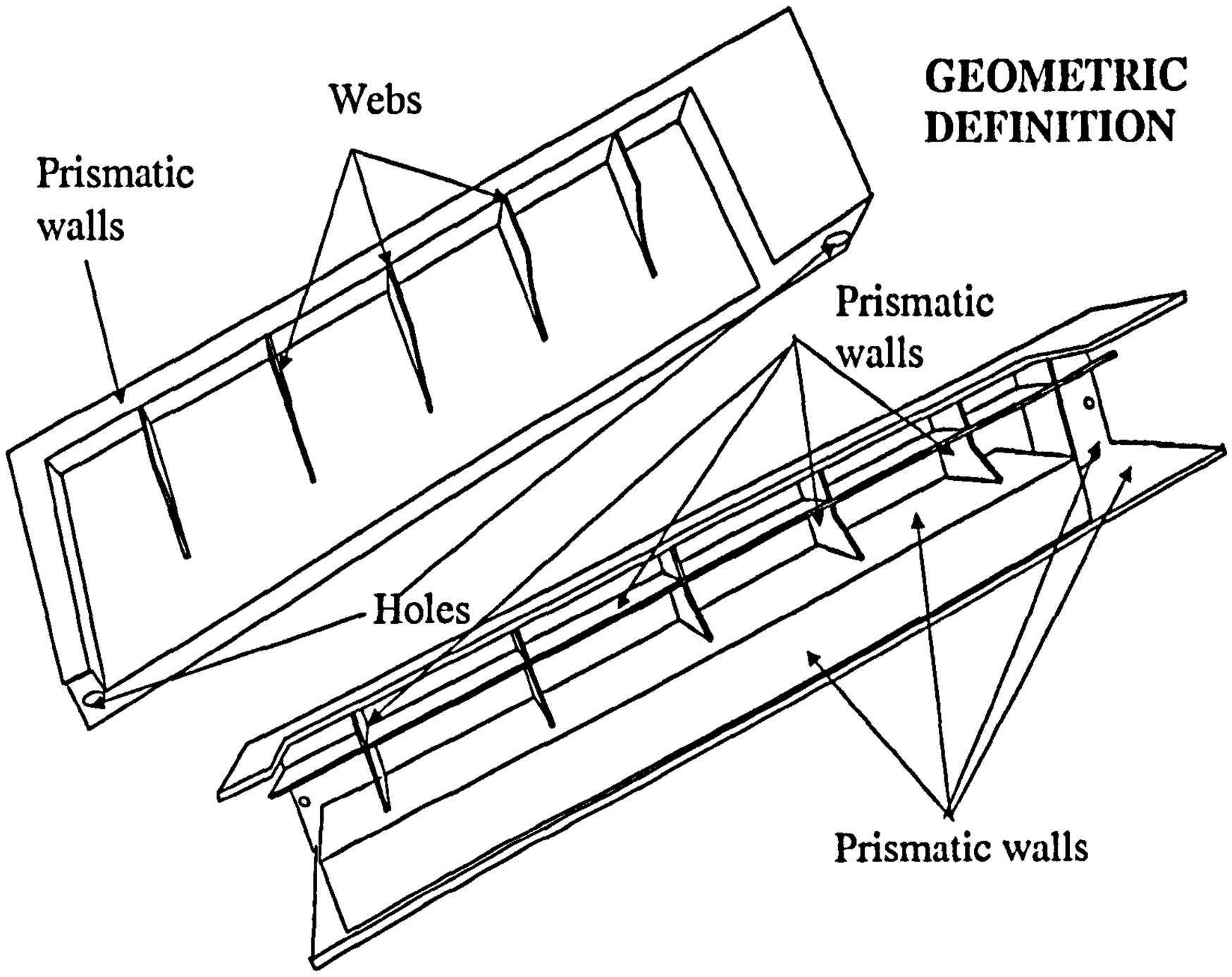


Figure 8.13 "Printer's component" plastic part design definition

Due to the knowledge integrity of the Manufacturing Knowledge Model, decisions such as which process parameters to use cannot be supported unless the manufacturability of the part's walls and holes has been ensured. The knowledge that belongs to the product engineer is shared for this purpose. In this case study, the "printer's component" part was designed within manufacturability constraints. Therefore, the process parameters selection constraints, presented in section 7.4.3, are invoked from the Manufacturing Knowledge Model by the Selection of Process Parameters application. These constraints calculate recommended process parameters, which are delivered in the following feedback advice (see the Selection of Production Equipment in figure 8.14):

- An injection velocity of 0.8 m/s, which could be increased to avoid weld lines.
- A recommended process temperature of 260.5 °C, which could be increased to avoid weld lines.
- A recommended injection pressure of 125,000,000 Pa, which can be increased to avoid weld lines. In this case study, it is assumed that the injection machine has already been selected, as shown in the Product Model of figure 8.14. By sharing and

providing the product life cycle data related to the selected machine the recommended injection pressure is suitable to the maximum injection pressure provided by the machine.

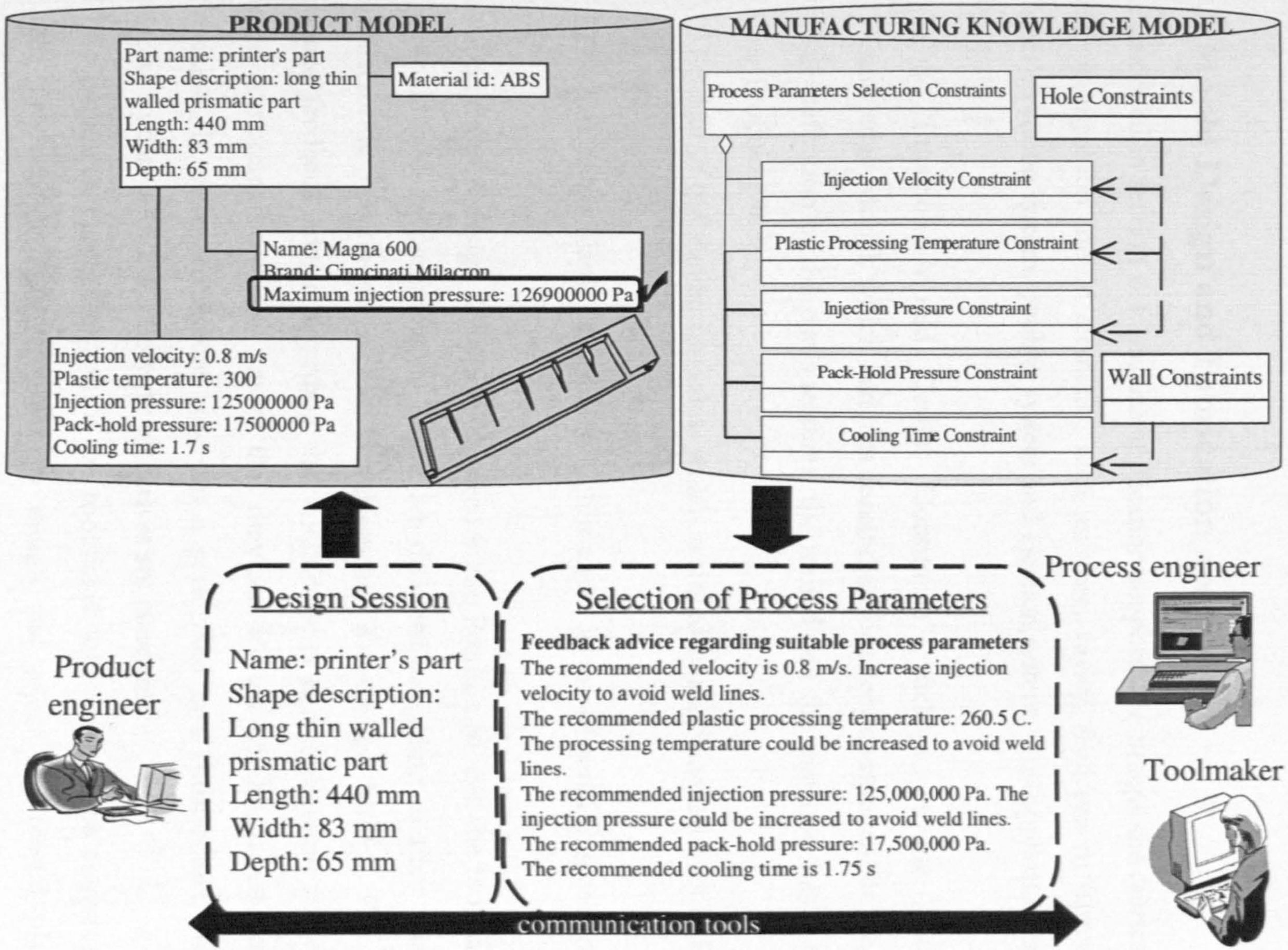


Figure 8.14 A scenario of collaborative selection of suitable process parameters for production

- A pack hold pressure of 17,500,000 Pa.
- A cooling time of 1.75 s.

Based on this feedback advice, the toolmaker selects the process parameters and stores them in the Product Model (see figure 8.14).

8.5 Mould Design and Fabrication application

As described in section 6.4.1.5, this application supports the design and fabrication of the different injection mould elements, such as: core, cavity, feed system (sprue, gate and runner), venting system, cooling system and ejection system. This application is supported by:

- The "Injection Mould Design Constraints" package of the Manufacturing Knowledge Model, which includes constraints for every component of the mould.
- The definition of the part's features, the mould and the injection machine stored in the Product Model.
- The data of the standard moulds, which is stored in the Standard Mould Engineering Model.
- The data of the plastic materials, which is stored in the Material Engineering Model.

Based on the part design definition captured in the Product Model, the toolmaker designs the mould and its components. These are designed according to their corresponding manufacturing constraints in the Manufacturing Knowledge Model. The constraints produce feedback advice regarding the suitable types and dimensions for each of the mould components and also ensure that they are defined within process and resources constraints. In order to explore how this is performed in a collaborative environment, two case studies with the following objectives are presented:

1. To explore the collaboration of the toolmaker with the process engineer as well as the product engineer in order to design and select the mould plates for an insert/bolster type of mould.

2. To explore the collaboration of the toolmaker with the product engineer in order to design the gating, ejection and venting systems of the mould.

8.5.1 Case study 6: collaborative design of the injection mould plates

Based on the part design definition, the toolmaker designs the mould plates for an insert/bolster type of mould. As explained in section 7.4.4.1, this type of mould requires a standard mould where the cavity and core plates are going to be inserted. The design of these plates could be started without the need to have a detail definition of all the features of the part in the Product Model. In such a way, the mould design could be performed concurrently to the part design and the selection of production equipment by having a minimum definition of part data.

The case study is done using the liquid container cap plastic part definition (see figure 8.5), which is assumed to be within manufacturability constraints. Furthermore, it explores how the collaboration is supported by a common source of knowledge and data during the following scenarios:

- The consideration of the mould plates
- The selection of a suitable standard mould

During the design of the mould plates, the collaboration between the toolmaker, product engineer and process engineer is essential. This is because the partial part definition needs to be considered as well as the injection machine, which is going to be used for production, in case this has already been selected.

8.5.1.1 Calculation of the suitable mould plates dimensions based on manufacturing constraints

The mould plates' dimensions (length, width and thickness) are calculated based on the definition of the plastic part stored in the Product Model, the number of cavities and their layout. For the selection of a number of cavities and its layout it is necessary the

collaboration of the toolmaker, the process engineer and the product engineer. This can be achieved in any of the following ways:

1. By the toolmaker, the process engineer and the product engineer using communication tools to discuss and decide on the number of cavities and its layout.
2. By the toolmaker deciding on the number of cavities and its layout. This data is then stored in the Product Model, so it is available to the process engineer and the product engineer for approval. Furthermore, the data can be used in other engineering applications, such as selecting an injection machine.
3. By the process engineer selecting the injection machine that is going to be used during production. The characteristics of this machine are stored in the Product Model and they are used to calculate a suitable number of cavities for the mould.

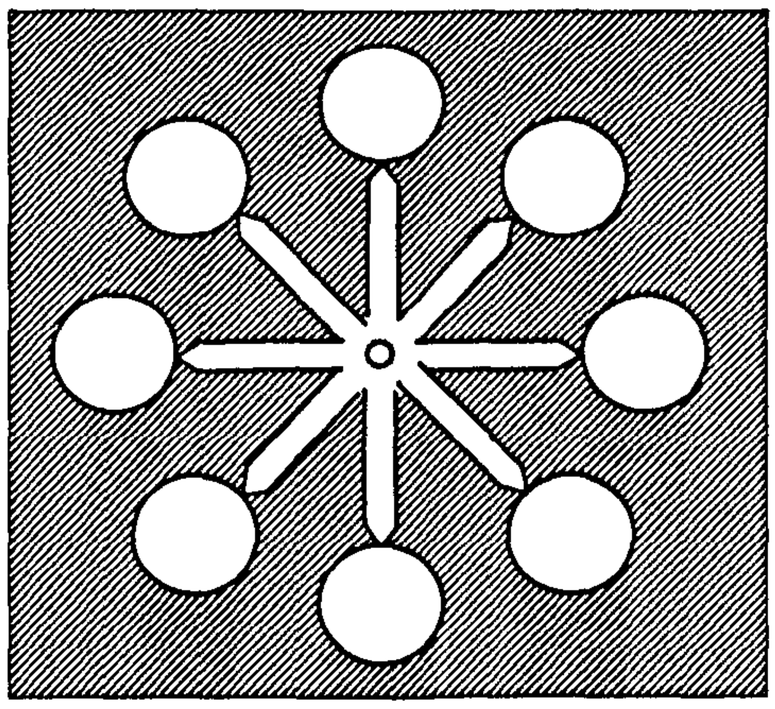


Figure 8.15 Balanced cavity arrangement for injection mould of case study 6

In this case study, the toolmaker collaborates with the process engineer and the product engineer to agree in having 10 cavities in a balanced arrangement type 1 (see figure 8.15). After accessing from the Product Model the part design definition, the following constraints for the mould plates are invoked:

- Injection Mould Plate Length Constraint
- Injection Mould Plate Width Constraint
- Injection Mould Thickness Constraint

As illustrated in the Mould Design application in figure 8.16, these constraints calculate a suitable plate length of 140 mm, width of 140 mm and mould thickness of 60 mm.

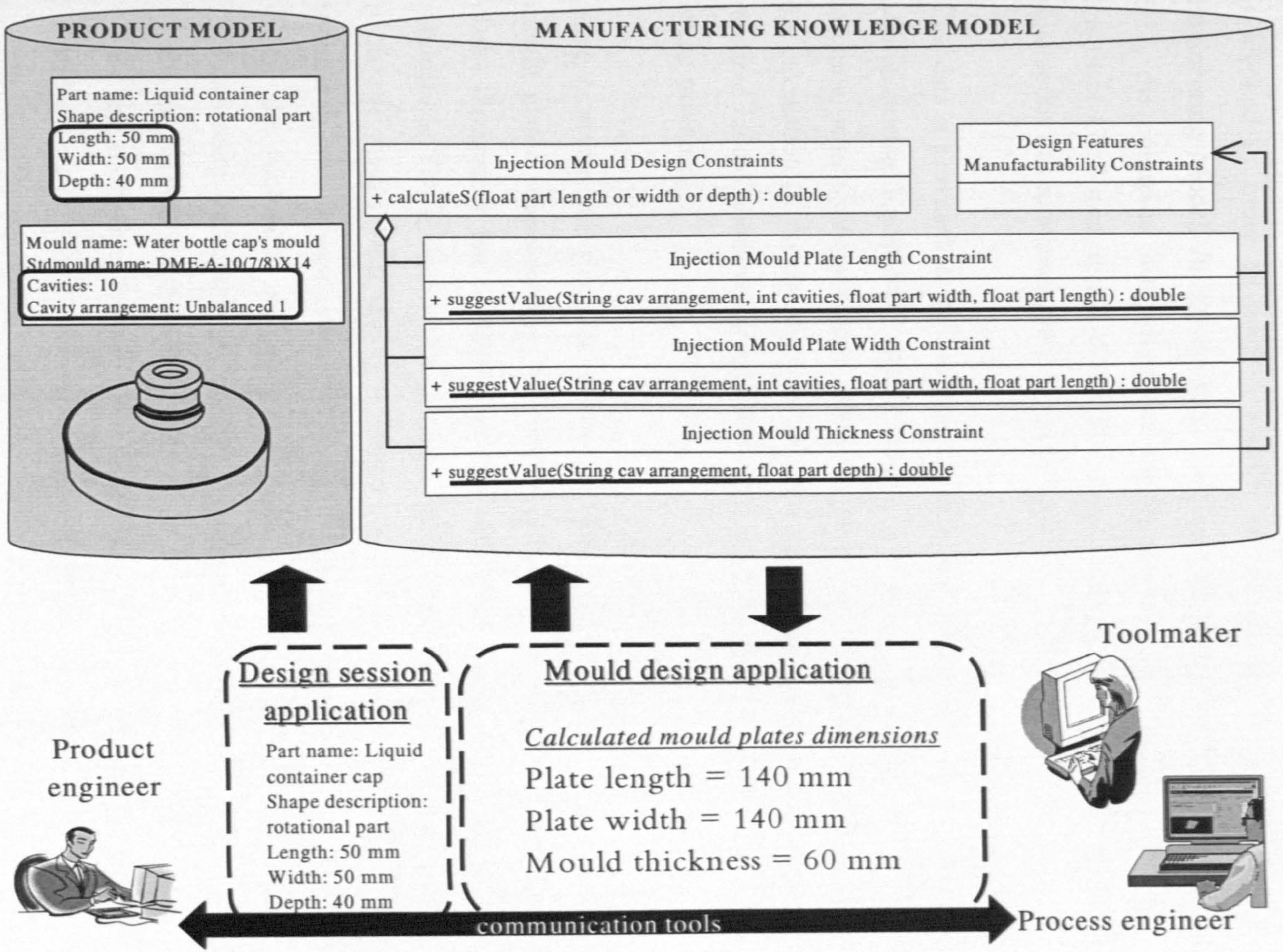


Figure 8.16 A scenario of collaborative design of mould plate dimensions

8.5.1.2 Selection of a suitable standard mould for the mould plates

After the mould plate dimensions have been calculated, the "Standard Mould Constraint" is invoked in order to select a suitable standard mould from the Standard Mould Engineering Model. As illustrated in the Standard Mould Engineering Model in figure 8.17, the constraint determines that the standard mould "DME-R-RB-6x7", with a length of 177.8 mm, width of 152.39 mm and average mould thickness of 153.98 mm is suitable for this rotational part's injection mould.

This data is provided to the toolmaker as feedback advice. Due to the integration of the Manufacturing Knowledge Model, these dimensions are suitable to fit the injection machine, which was previously selected by the process engineer. This is because during the calculation of the suitable machine, the knowledge regarding the calculation of the required mould width and length was used to calculate the required space for the mould in the machine.

After receiving the feedback advice, the toolmaker modifies or stores the suggested mould plates' dimensions in the Product Model. However, modifying the suggested plate's dimensions has an impact on the injection machine that has been selected for production. This is because it is possible that the machine would not be suitable anymore to fit the mould with the new dimensions. This is immediately highlighted to the toolmaker, who collaborates with the process engineer in either of the following ways:

- By the toolmaker and process engineer concurrently accessing the Selection of Production Equipment application in order to ensure that the machine selected fits the mould with the new dimensions.
- By the toolmaker accessing the Selection of Production Equipment application in order to determine if the injection machine selected is suitable for the mould with the modified dimensions. If this is the case, the toolmaker can continue defining the mould. Otherwise, the toolmaker selects another machine. For this, product life cycle knowledge related to the selection of a suitable injection machine is transparently invoked and feedback advice produced.

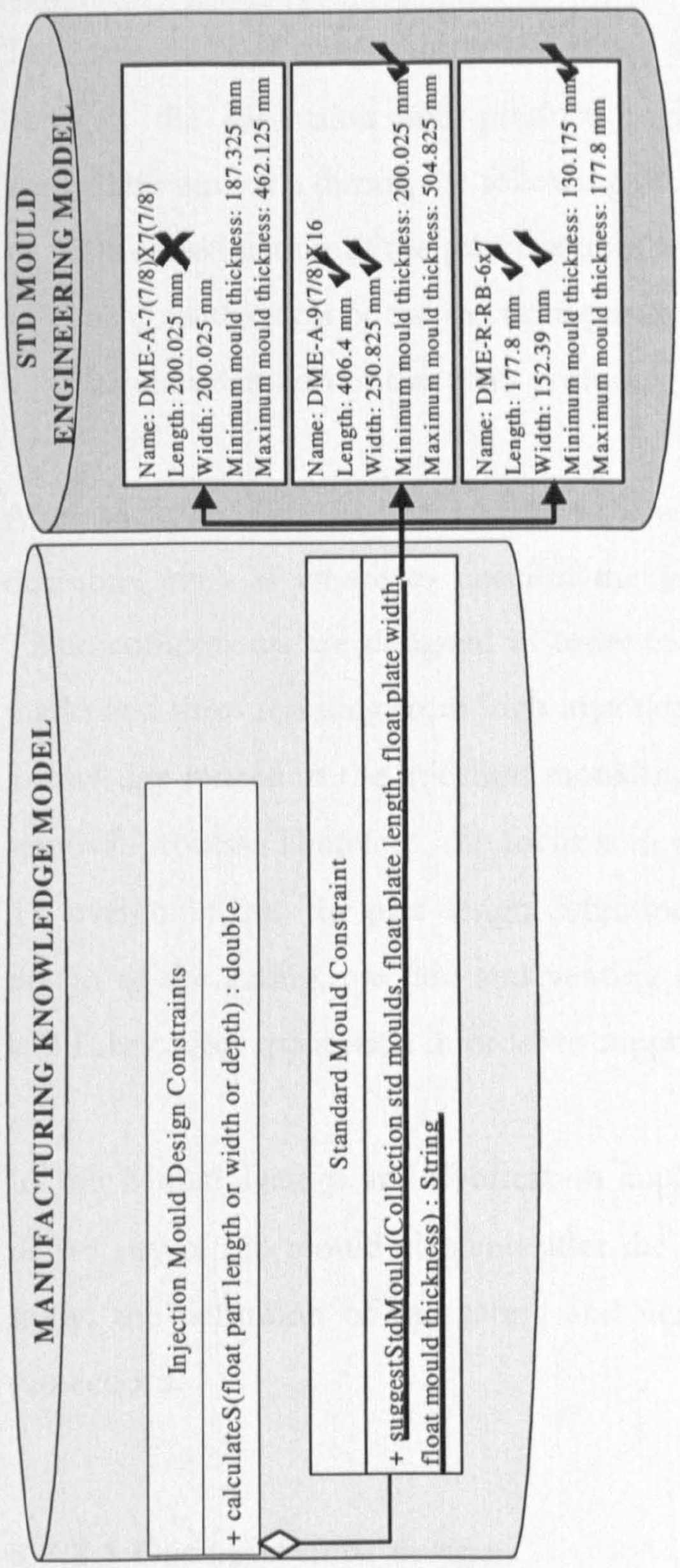


Figure 8.17 A scenario of selection of suitable standard mould dimensions

The experimentation of this case study in the implemented software prototype is presented in section 11.5.1.

8.5.2 Case study 7: collaborative design of the gating, ejection and venting system of the injection mould

This case study is using a batteries' cover plastic part and it explores the collaboration between the toolmaker and product engineer supported by a common source of knowledge and data during the following scenarios:

- The consideration of the gate positions and types
- The consideration of the ejections positions and types
- The consideration of the vents positions

After the part data has been defined, the toolmaker designs the mould and makes decisions, such as where to position the gates, ejection pins and vents in the mould. These components are designed in order to avoid short shots, weak welding lines, burn marks and stress resulting from high injection pressure. Although the toolmaker has some knowledge related to the injection moulding process, his expertise is mainly in the metal removal process. Therefore, the focus is in designing a mould which is easy to fabricate. To overcome this, the part design definition is shared and the knowledge regarding the design of the gating, ejection and venting system is invoked during the Mould Design and Fabrication application in order to support the collaborative design of the mould.

In the Mould Design and Fabrication application, the toolmaker has the flexibility to define any of the mould elements after the mould plates have been defined. In this case study, the definition of the gating and venting system are presented in the following subsections.

8.5.2.1 Gating system design

The application forces the toolmaker to consider the gating system before the vents. This is due to an interaction between the vents position and the gating system constraints (see section 7.5.1), as the vents position depend primarily on the gate(s) position. This advice is displayed by the application in case the toolmaker attempts to design first the venting system.

In order to position the gates, the toolmaker needs to collaborate with the product engineer to agree on the best position. For this, the constraint related to the suitable gate(s) position ("Gating System Position Constraint", see section 7.4.4.5.2) is invoked during the design activity. The constraint detects there are two bosses (see the Product Model in figure 8.18) and recommends the farthest away point as a candidate gate position. The advice produced is delivered to the product engineer by the Design for Manufacturing application. By using this advice, the product engineer and toolmaker collaborate to make a final decision by either of the following approaches:

- The product engineer selects and stores in the Product Model a candidate gate position. This position is later considered by the toolmaker during the mould design.
- The product engineer and toolmaker collaboratively agree on a suitable position with the use of a communications mechanism.

In this case study, the toolmaker and product engineer agree on the position (25,40,0) of wall3 as a suitable gate position (see the Design for Manufacturing application in figure 8.18). Other decisions such as which type of gate to use and dimensions of the gate are taken by the toolmaker, during the Mould Design and Fabrication application, based on the following manufacturing constraints:

- Gating System Type Constraint
- Edge Gate Width Constraint
- Edge Gate Depth Constraint
- Edge Gate Land Length Constraint

As illustrated in the Mould Design application in figure 8.18, these constraints produce feedback advice recommending to use an edge gate with a land length of 0.625 mm, a width of 1.47 mm and a depth of 1.39 mm and. Based on this advice, the toolmaker defines the edge gate by its attributes (i.e. name, position, wall where the gate is placed, land length, width and depth).

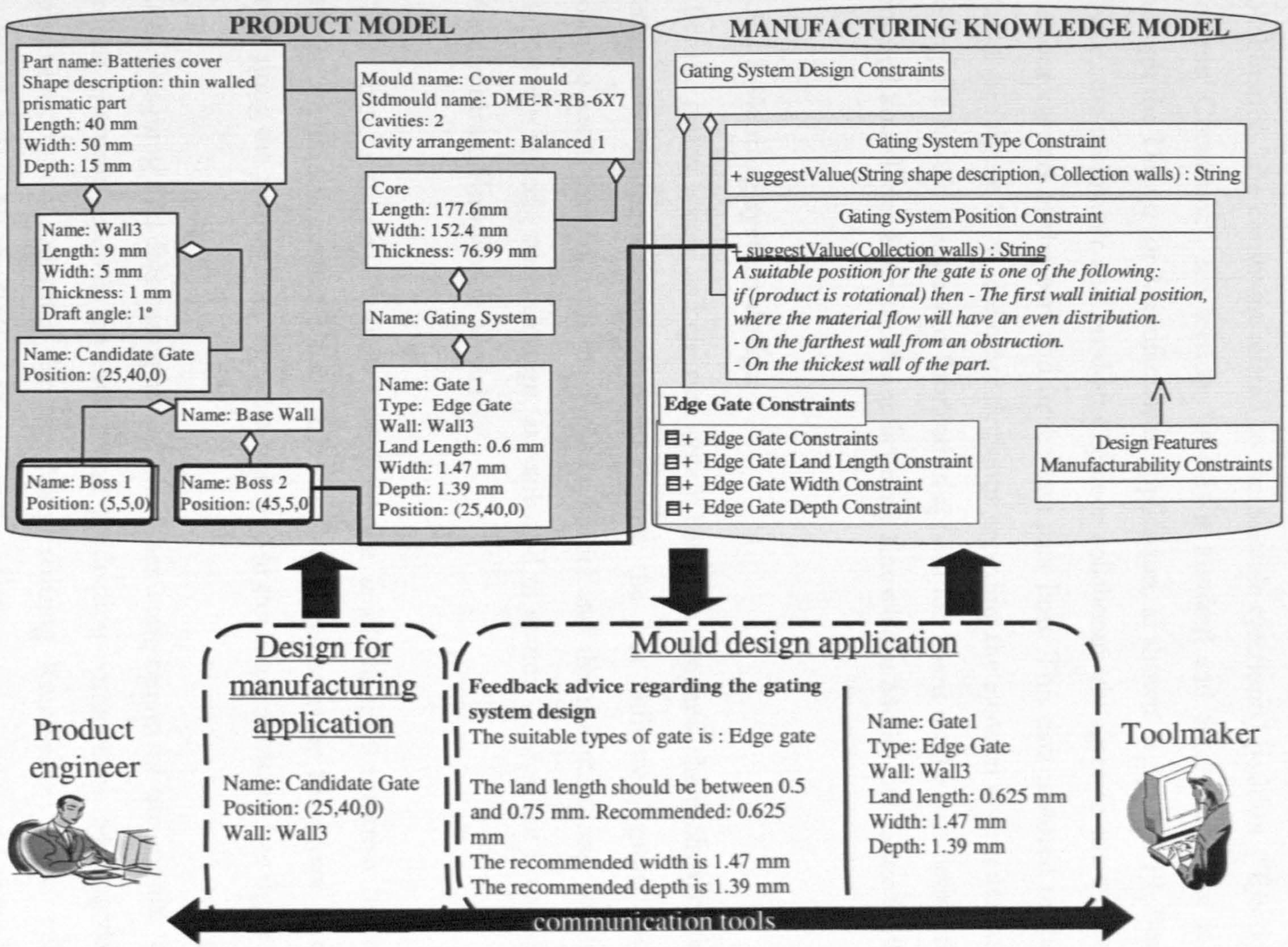


Figure 8.18 A scenario of collaborative design of the gating system in the mould

8.5.2.2 Ejection system design

The ejection system should also be designed before the venting system as described in section 7.4.4.4. In a similar case than for the gates, the toolmaker needs to collaborate with the product engineer to agree on the best positions for the ejections. To support this collaboration, the constraint related to the suitable ejection(s) position ("Ejection System Position Constraint", see section 7.4.4.3) is invoked and feedback advice is delivered through the Design for Manufacturing application, as shown in figure 8.19. Based on the advice, the toolmaker and product engineer collaboratively agree to use ejection pins in the four corners of the part and in the complex boss. This data is stored in the Product Model (see figure 8.19) and other decisions regarding the ejection system design are made during the Mould Design and Fabrication application based on the part definition and the required knowledge from the Manufacturing Knowledge Model (see figure 8.19).

8.5.2.3 Venting system design

After the gating system and ejection system has been designed, the toolmaker proceeds to design the venting system. To support this, the part definition, gating and ejection positions are retrieved from the Product Model and the manufacturing constraints that support the venting system design, as explained in section 7.4.4.4, are invoked from the Manufacturing Knowledge Model.

These manufacturing constraints produce the feedback advice shown in the Mould Design application in figure 8.20. Based on this feedback, the toolmaker places several vents along the parting line. This data is stored in the Product Model (see figure 8.20).

After designing and fabricating these and other components of the mould, the plastic process engineer receives the mould and production commences. Making decisions in collaboration and supported by the Manufacturing Knowledge Model ensures that problems caused by the lack of consideration of process, resources and other limitations by both the product engineer and toolmaker will not arise anymore during the production stage, which will eliminate further interactions. Section 11.5.2 presents the experimentation done in the implemented software prototype using this case study.

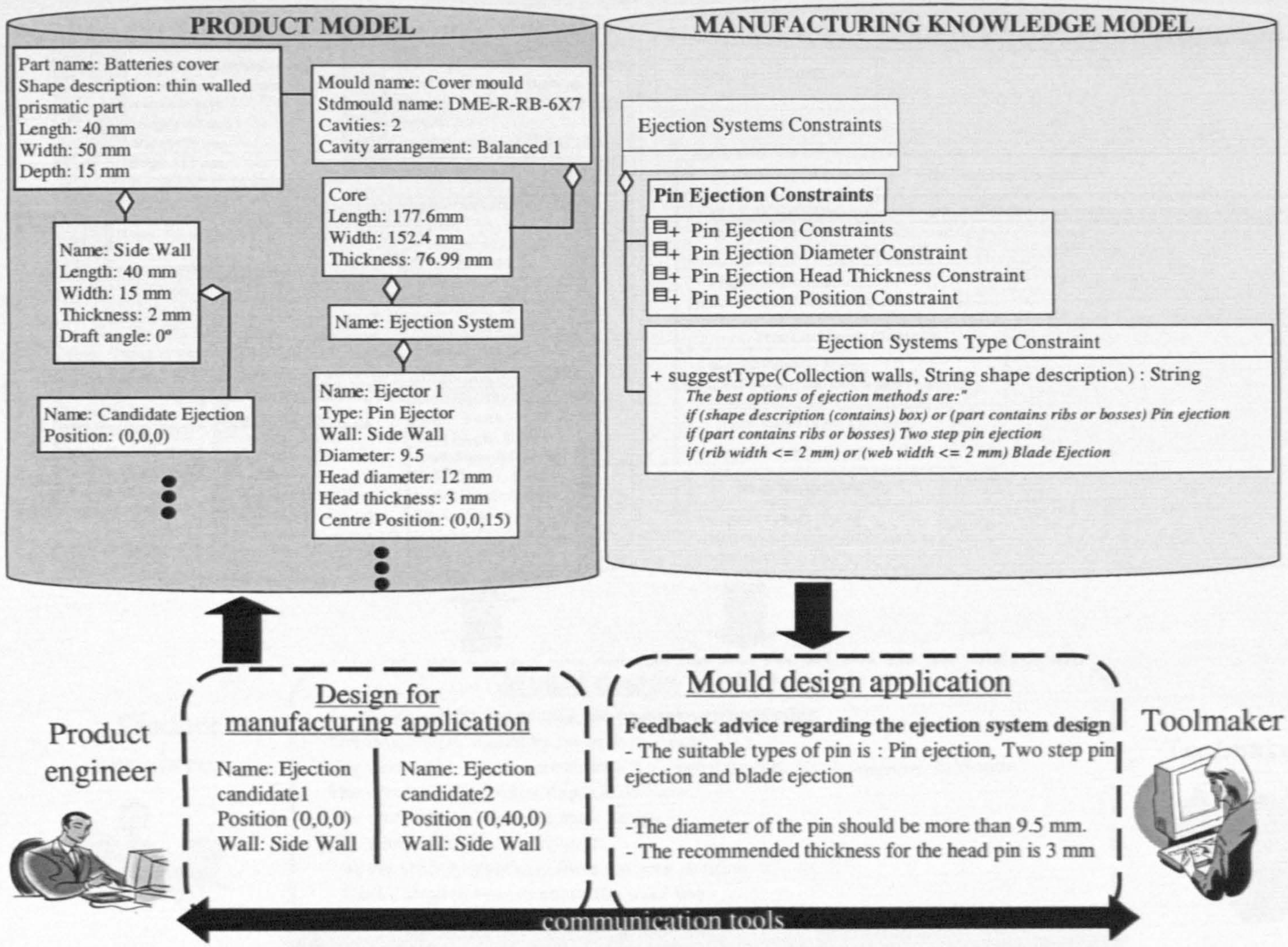


Figure 8.19 A scenario of collaborative design of the ejection system in the mould

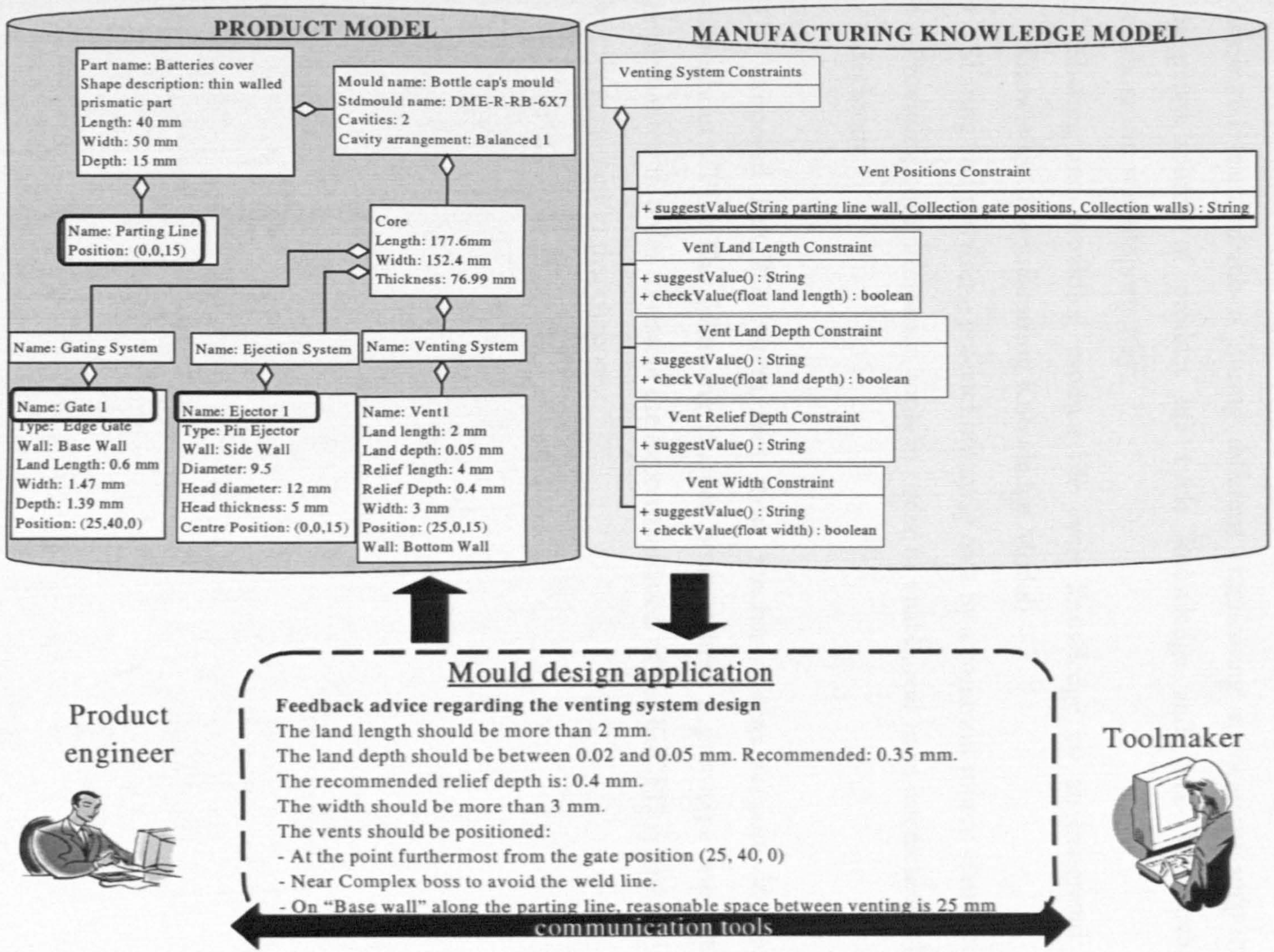


Figure 8.20 A scenario of collaborative design of the venting system in the mould

8.6 Closing remarks

The case studies presented in this chapter explored the collaboration of the geographically distributed team members during different engineering applications supported by an integrated source of product life cycle knowledge and data. In conclusion, the collaboration was supported by:

- Sharing and providing product life cycle knowledge by an integrated source of knowledge (Manufacturing Knowledge Model)
- Sharing and providing product life cycle data by a common source (Product Model)
- Providing communication tools in order to enable real time interaction when taking decisions.

The proposed KdCPD system uses these mechanisms to support decision making throughout CPD. The following chapter presents how a prototype was designed and implemented in order to enable the experimentation of the KdCPD system using the case studies presented in this chapter.

Chapter 9

Object Oriented Design of the KdCPD System

9.1 Introduction

This chapter presents the object oriented design of a prototype of the KdCPD system architecture proposed in chapter 6. Section 9.2 presents the objectives of the design and implementation of this prototype, while the object oriented design stage is explained in section 9.3. This chapter concludes with closing remarks in section 9.4.

9.2 Objectives of the object oriented design and implementation of the KdCPD system

A prototype of the KdCPD system architecture, presented in chapter 6, was designed and implemented using object oriented enabling techniques. This was done in order to demonstrate the main innovation of the research, in addition to the following specific objectives:

1. To demonstrate how the proposed KdCPD system architecture could be implemented in an object oriented environment as a functional software prototype to support the collaboration between the geographically distributed team members.
2. To demonstrate the implementation of the geographically distributed product life cycle knowledge in the Manufacturing Knowledge Model.
3. To demonstrate and validate how the distributed product life cycle data and knowledge could be provided through the KdCPD system engineering applications to support CPD.

The following sections describe in more detail how the prototype was designed in order to achieve these objectives.

9.3 Object oriented design of the KdCPD system architecture

The KdCPD system architecture, presented in section 6.2, was formally designed using an object oriented technique following the CIMOSA reference framework. By using this technique, the elements of the system are seen in terms of 'entities' instead of operations and functions (Sommerville 2001). Designing the system in such a way provided the following advantages:

- It facilitated the unambiguous identification and formal representation of each of the elements of the system as well as their interactions.
- It provided a blueprint for implementing the KdCPD system prototype using object oriented enabling technologies.
- It facilitated the future maintenance and upgrading of the prototype, as each of the elements of the system was viewed as an independent entity, which can be implemented and modified without affecting other elements of the system.

Several object oriented design methodologies have been proposed (Booch 1994; Coad and Yourdon 1991; Coleman et al. 1994; Jacobson 1992). More recently, a unification of the notations used in these methodologies has been defined (Unified Modelling Language) along with an associated design process (Rumbaugh et al. 1999). All these methodologies aim to capture the design decisions by modelling every element of the system using different perspectives. This research used the UML notation and followed a simplified methodology, which incorporates activities that are common to most object oriented design methodologies. In this respect, it is comparable to the proposed UML design process (Rumbaugh et al. 1999). Hence, the following activities were conducted:

1. Overall design of the KdCPD system architecture: this activity has been described in chapter 6.
2. Representation of the elements of the KdCPD system using UML object oriented language. For this, the UML notation[1] used is the following:

[1] For a detail description of the UML language refer to Appendix D.

- A class is a set of objects that share a common structure and a common behaviour while an object is an instance of a class. The definition of a class includes the declaration of its name, attributes and methods, as illustrated in figure 9.1-a. In the KdCPD system design, the information, knowledge and applications are represented using classes. For example, the features "Wall" (see section 7.6), the manufacturing constraint "Wall Thickness Constraint" (see section 7.4.1.1) and the application "Design for Manufacturing" (see section 6.4.1.3) are all considered as classes. These classes contain attributes and methods that describe the behaviour of the class. For example, the "Wall" class contains attributes, such as thickness, direction and draft angle. The "Wall Thickness Constraint" class contains methods, which capture the manufacturing constraint for the wall thickness, and the "Design for Manufacturing" application contains methods that describe the functionality of the application.
- A package is a modular component containing two or more classes or subpackages, as shown in figure 9.1-b.

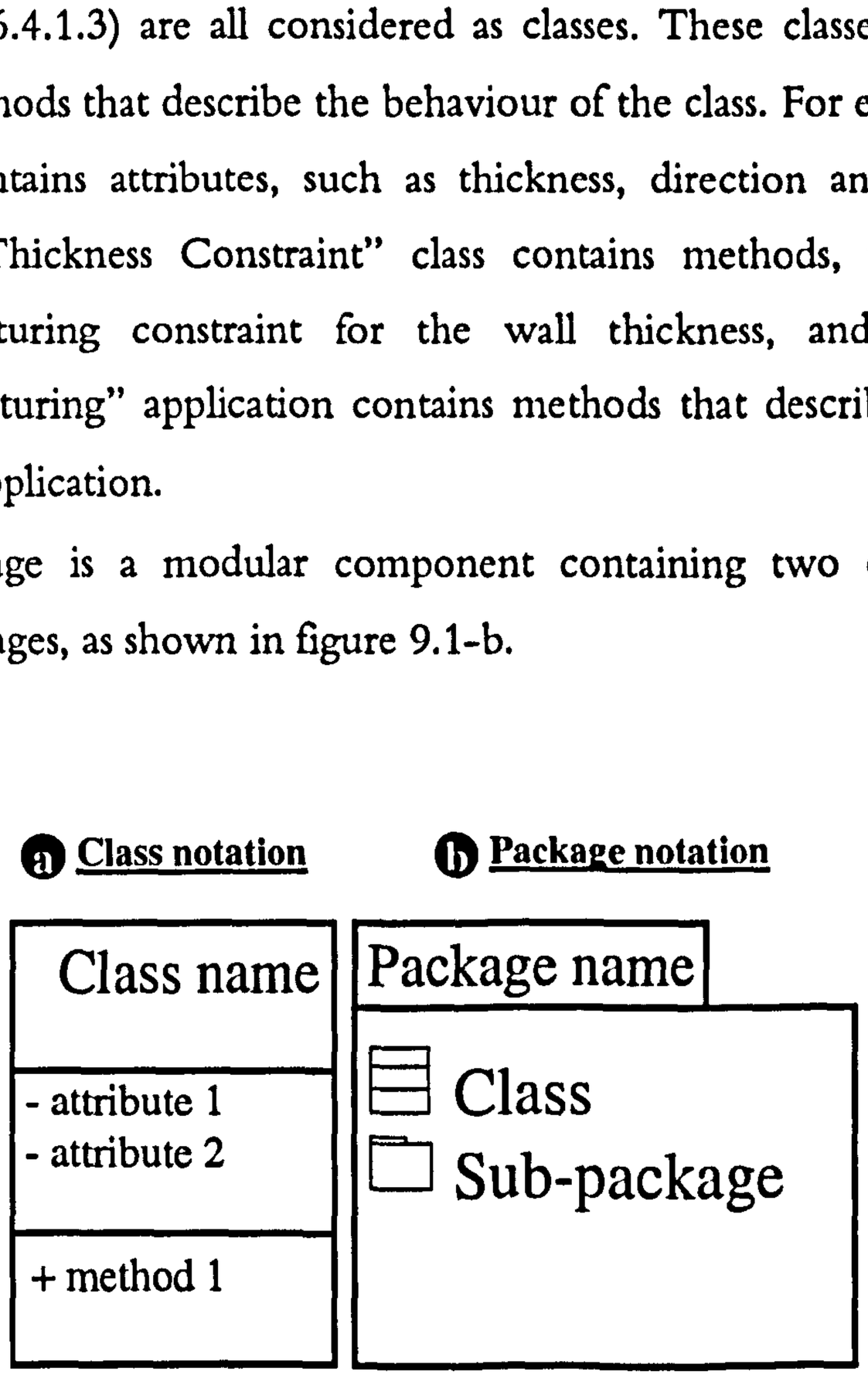


Figure 9.1 UML notation of a class and a package

3. Development of object oriented design models. The classes represented in the previous activity are static, while the objects that are instantiated from them are

dynamic. This means that the objects are created and destroyed during the execution of the system. For this, two types of design models were developed:

- Static design models: these show the classes of the system and their interactions.
- Dynamic design models: these show a snapshot of what might happen in an object's life during the execution of the system.

The following section presents in detail the final activity of the object oriented design process. This activity encompasses the representation of the system elements using UML notation.

9.4 Design models of the different elements of the KdCPD architecture

As described in chapter 6, the system architecture is composed of three layers: information, application, and end user layer. In order to represent the geographically distributed nature of the companies, the elements of each layer will be physically placed in geographically distributed locations. Therefore, the architecture was designed with a distributed object oriented client-server approach. In this approach, the elements are viewed as objects distributed across a number of computers on the network. These objects are regarded as clients or servers, which communicate through a middleware. In the KdCPD system design, the knowledge and information databases are considered to be servers as they provide services to the decision support engineering applications, which are considered to be clients. The middleware is called Object Request Broker (ORB) and its role is to provide a seamless interface between the objects (Sommerville 2001).

Figure 9.2 shows the top level static design model of the research concept illustrated in figure 6.1 of section 6.2. In this model, the different layers of the system architecture are composed of a set of packages as follows:

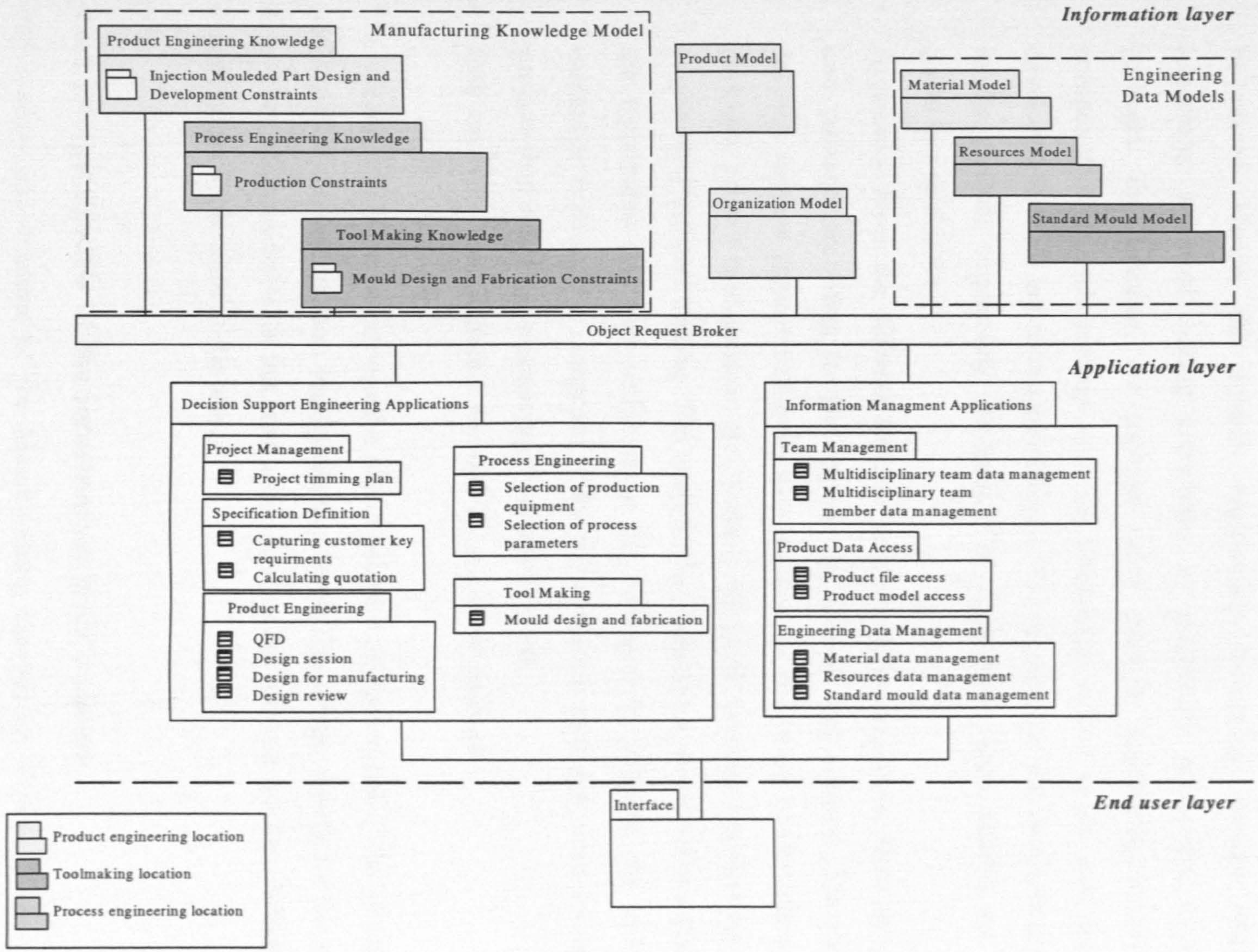


Figure 9.2 Top level of the design model of the KdCPD system architecture

- Information layer: the Manufacturing Knowledge Model and Engineering Data Models were broken down into packages according to the location where they will be placed. Hence, the product engineering knowledge, process engineering knowledge and tool making knowledge are graphically represented in different packages. For example, the package called product engineering knowledge is composed of one sub package with the knowledge related to the part design and development of an injection moulded part. This knowledge will be physically placed in the product engineering company. The Engineering Data Models are broken down in a similar way.
- Application layer: the different decision support engineering applications are grouped into packages according to the engineering activity they support. Therefore, the decision support engineering application package is composed of the following sub packages: project management, specification definition, product engineering, process engineering and tool making. The engineering applications, described in section 6.4.1 are represented as classes within these sub packages. In addition, the information management package is composed of the following sub packages: team management, product data access and engineering data management.
- End user layer: the interface is represented as a single package.

The following subsections present the object oriented design models of the elements that are located in the information and application layer. The design models for the end user layer were not considered, as this layer only refers to the Internet browser, which is used to present the application to the user.

9.4.1 Design models of the information layer's classes

The content and structure of the Manufacturing Knowledge Model, Product Model, Organisation Model and Engineering Data Models have already been represented using the object oriented UML notation in chapter 7. Therefore, these models were reused as static design models during the object oriented design of the KdCPD system. Additional classes were included to facilitate the implementation of the models in the software prototype. The following subsections describe in more details these design adjustments.

9.4.1.1 Representation of integrity of the Manufacturing Knowledge Model in the design model

In order to capture the common behaviour of all the manufacturing constraints of the Manufacturing Knowledge Model, an extra set of class were included in the design model. As such, the "Attribute Constraint" class, illustrated in figure 9.3, contains the following attributes:

- Name: captures the name of the constraint
- Description: captures a brief description of the manufacturing constraint in text.

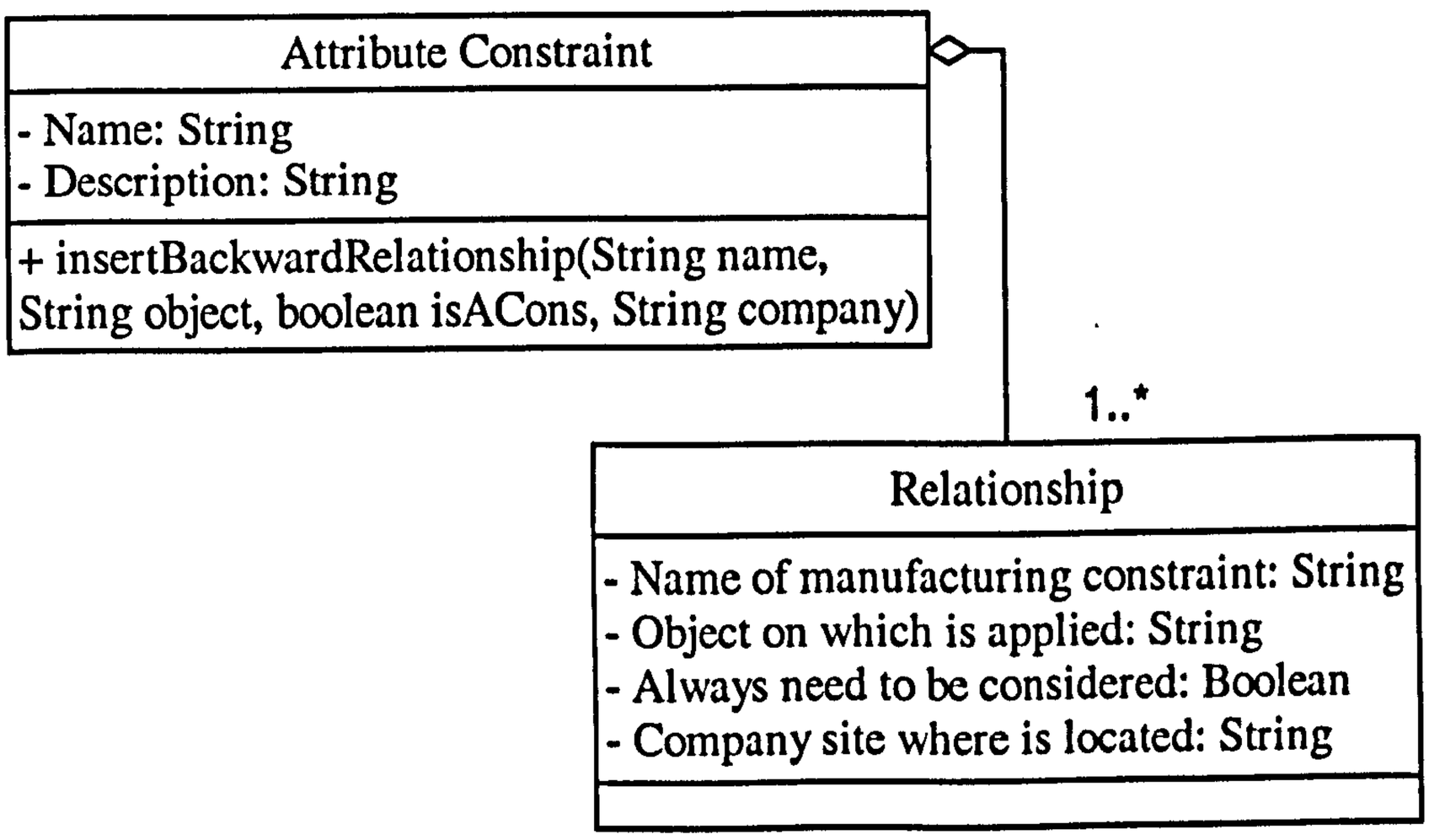


Figure 9.3 "Attribute Constraint" class representation in UML notation

The "Attribute Constraint" class also contains an aggregation relationship with the "Relationship" class (see figure 9.3). This means that an instance of the "Attribute Constraint" class may have related one or more instances of the "Relationship" class. The latter class represents the interactions among manufacturing constraints and it has as attributes:

- Name of the manufacturing constraint
- Object on which the constraint is applied: for example, "Rib Draft Angle Constraint" is applied on the rib.
- If the constraint needs to be always considered

- Company site where the constraint is located

In addition, the "AttributeConstraint" class has a method called "insertBackwardRelationship" for storing a relationship of the manufacturing constraint. This method receives as input parameters the name of the constraint, the object on which the constraint is applied, if it needs to be considered and the company where the knowledge is located.

An example of how the "Attribute Constraint" class is inherited is illustrated in figure 9.4. The "Vent Positions Constraint" object, which represents the rules described in section 7.4.4.4, inherits the name and description attributes from the "Attribute Constraint" class. Furthermore, there is an aggregation relationship with the following "Relationship" objects:

- Relationship with the constraint "Gating System Design Constraint", which is applied on the gating system and needs to be considered at all times. This means that the gating system must have been designed in order to consider the venting positions. The object also indicates that the knowledge is located in the tool making site.
- Relationship with the constraints: "Reinforcement Draft Angle Constraint", "Reinforcement Base Radius Constraint" and "Reinforcement Distance Constraint". These constraints are applied on the reinforcement and are not required to be considered, which means that it is not necessary to have a reinforcement on the part to consider the venting system position. However, if the part did contain any reinforcements, these would need to be within manufacturing constraints. These constraints are located in the product engineering site.

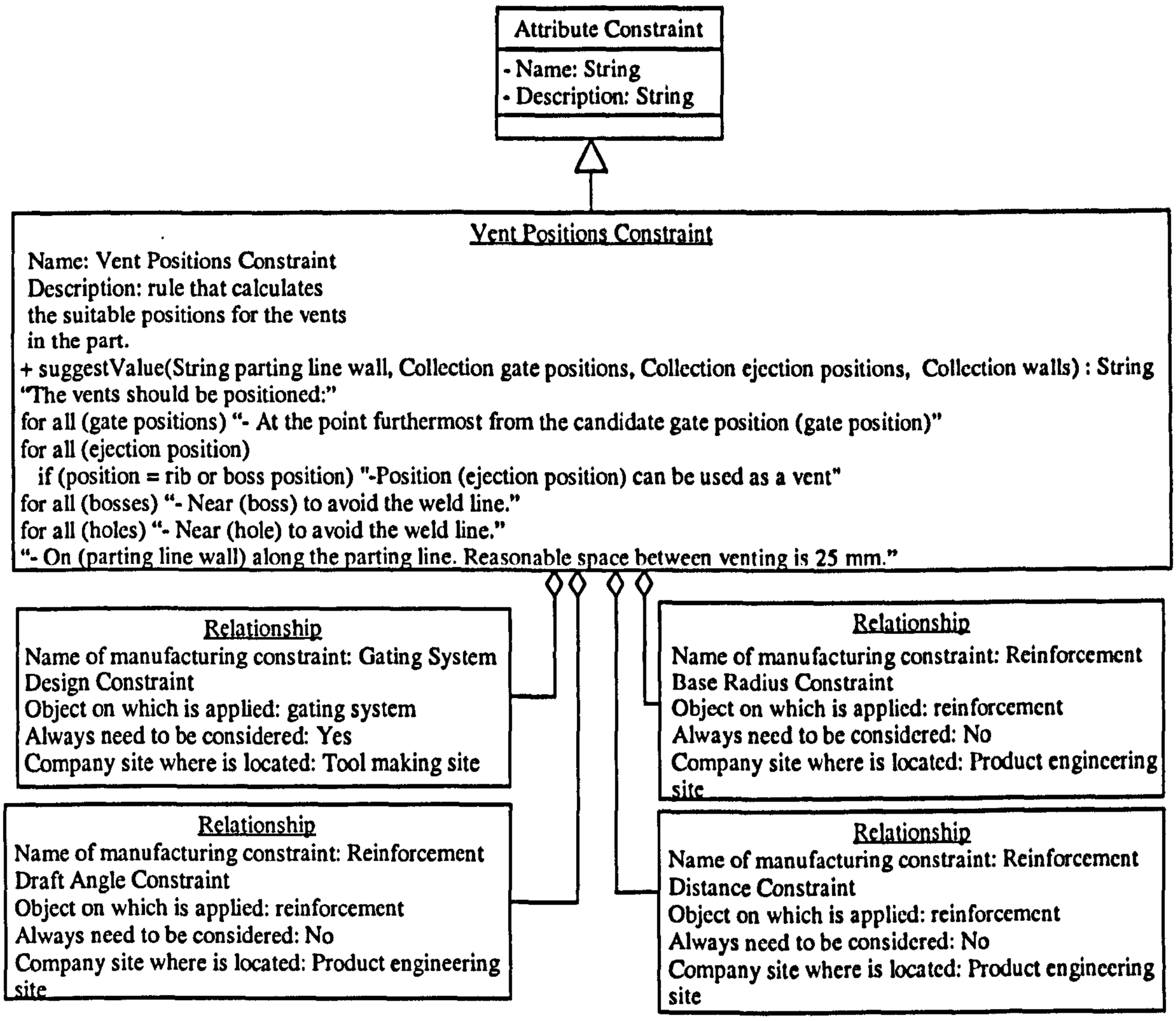


Figure 9.4 "Vent Positions Constraint" object inheritance from the "Attribute Constraint class in UML notation

9.4.1.2 Representation of the geographical distribution of the knowledge and information in the design model

Table 9.1 illustrates how the knowledge and information was geographically distributed among the product engineer, toolmaker and process engineer. These partner's locations were identified during the activity modelling in section 5.3 and were highlighted in the Manufacturing Knowledge Model in section 7.5.

Table 9.1 Geographically distributed location of product life cycle information and knowledge packages in the KdCPD system architecture

Location	Product life cycle information and knowledge
Product Engineering site	• "Injection Moulded Part Design and Development Constraints" package of the Manufacturing Knowledge Model • Product Model package • Organisation Model package • Material Model package
Process engineering site	• "Production Constraints" package of the Manufacturing Knowledge Model • Resource Model package
Tool making site	• "Mould Design and Fabrication Constraints" package of the Manufacturing Knowledge Model • Standard Mould Model package

During the implementation stage of the KdCPD prototype, the Manufacturing Knowledge Model, Product Model, Engineering Data Models and Organization Model were implemented as databases, containing the product life cycle information and knowledge to support the engineering applications. These applications access the databases through the middleware (ORB) by using a manager class, which is designed and implemented within the information layer. This manager class has the purpose of gaining access to the knowledge and data stored in the databases and of managing their storage, update and removal

In the KdCPD prototype, one manager class was designed and implemented for each location of the geographically distributed partners. As such, figure 9.5 illustrates the static design model of the manager classes. In this model, the "Product Engineering Manager", "Process Engineering Manager" and "Tool Making Manager" classes are the interface between the ORB and the information and knowledge models.

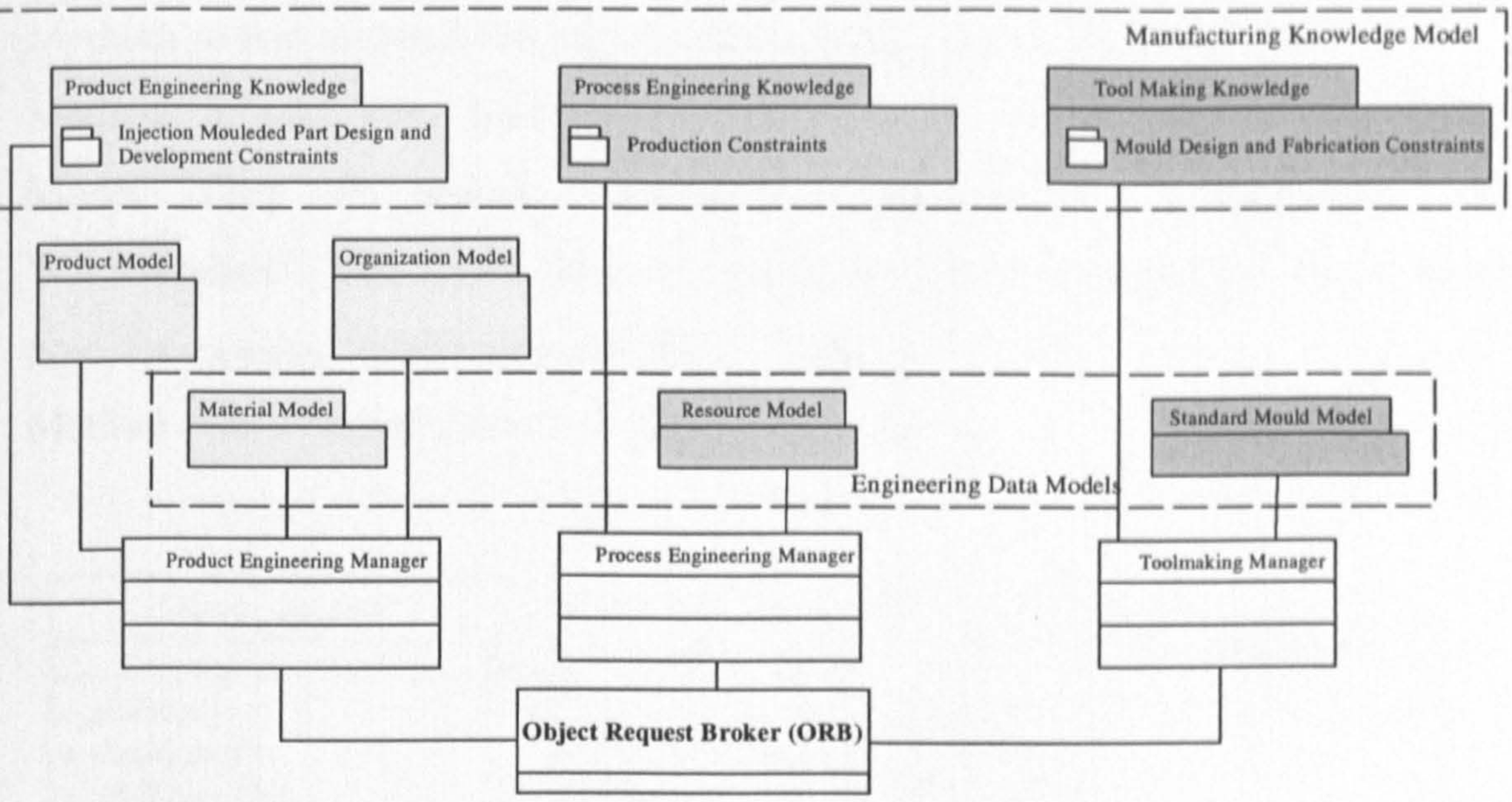


Figure 9.5 The design model of the manager classes of the information layer

9.4.2 Object oriented design models of the application layer's classes

The subsequent subsections describe the object oriented design models of the classes that represent the engineering applications: Design Session, Design for Manufacturing, Selection of Production Equipment, Selection of Process Parameters and Mould Design and Fabrication. Both static and dynamic design models are described in order to understand the interactions of the classes, which represent applications, information and knowledge, and their instances during the execution of the system.

9.4.2.1 The object oriented design model of the "Design Session" class

The main functionality of this class, described in section 6.4.1.3, is to capture the part definition in terms of features, such as walls, ribs and webs. The geometric representation of the part is generated in a 3D virtual model and the definition is stored in the Product Model.

Static object oriented design model of the "Design Session" class

Figure 9.6 illustrates the "Design Session" class and their relationships with other classes of the system. The methods, which capture the behaviour of the class are:

- Methods to start and end the object, called "startup" and "shutdown".
- Methods to add, edit, delete and retrieve feature's definitions from the Product Model. They are named "addFeature", "editFeature", "deleteFeature", and "accessFeature". The word "Feature" could be replaced depending on the feature type, for example "addPrismarticWall" or "editRib".
- Method called "drawFeature" to display 3D geometry.

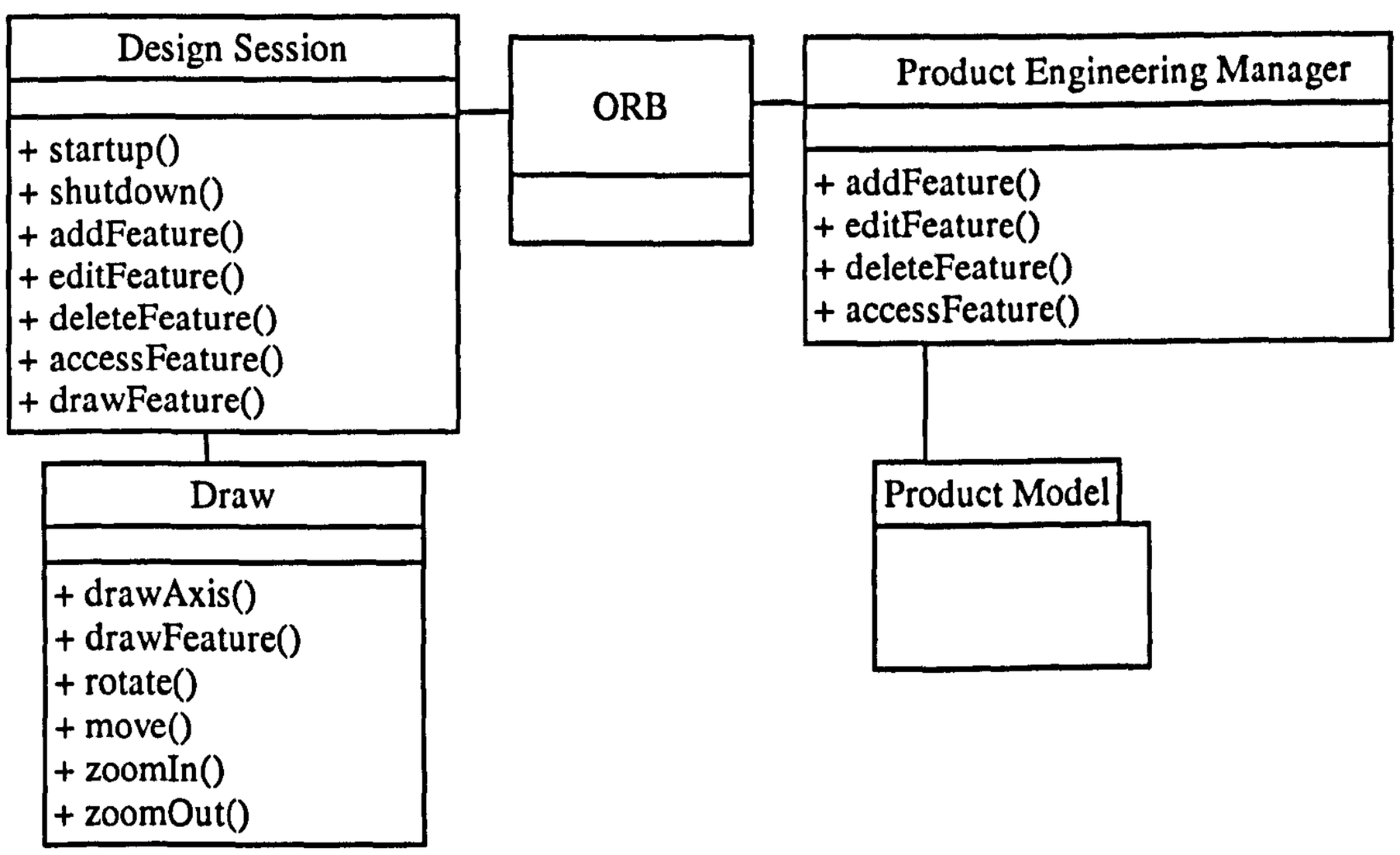


Figure 9.6 The static design model of the "Design Session" class

Furthermore, the "Design Session" class has an association relationship with the "ORB" class as well as with the "Draw" class. The "ORB" class represents the interface to the "Product Engineering Manager" class. This manager adds, edits, deletes and retrieves the feature's definition from the Product Model package. In addition, the "Draw" class addresses the functionality of generating a 3D virtual geometric model. As such, it contains methods to draw a feature, rotate it, move it and zoom it.

Dynamic object oriented design model of the "Design Session" class

Figure 9.7 illustrates the interactions between the objects that represent the Design Session application. During the execution of the system, these are instantiated from the

classes represented in the "Design Session" static model. In order to describe a scenario where the system is being executed case study 1 (see section 8.2.1) is used.

Figure 9.7 illustrates how the product engineer, being the end user, inputs the definition of "Base Wall" through the "Design Session" object (see figure 9.7-1). This object then triggers the "addPrismaticWall" method to request the "Product Engineering Manager" to store the feature definition (see figure 9.7-2). This request is done through the ORB object. The "Product Engineering Manager" object then executes the method called "addPrismaticWall" and stores the "Base Wall" definition in the Product Model database (see figure 9.7-3). It then returns a response regarding the successful storage (see figure 9.7-4). Upon receiving the response, the "Design Session" object requests the "Draw" object the generation of a 3D virtual model of "BaseWall" by triggering the "drawPrismaticWall" method (see figure 9.7-5). Thereafter, the "Draw" class generates the 3D virtual model (see figure 9.7-6). The "Design Session" object then displays a feedback to the end user, informing about the successful storage of "Base Wall" (see figure 9.7-7).

9.4.2.2 The object oriented design model of "Design for Manufacturing" class

The main functionality of this class is to ensure that the part's features can be moulded without problems (see section 6.4.1.3). The class also provides feedback advice whenever problems arise.

Static object oriented design model of the "Design for Manufacturing" class

Figure 9.8 illustrates the static model, where the "Design for Manufacturing" class contains the following methods that capture the behaviour of the class:

- Methods to start and end the object, called "startup" and "shutdown".
- Method to perform the design for manufacturing analysis, called "startDFM"
- Method to advice where weld lines are expected, called "suggestWeldLine"
- Method to suggest suitable gates positions in the part, "suggestGatePositions"
- Method to suggest suitable ejection pins positions in the part, "suggestEjectionPositions"

- Methods to add, edit, delete or retrieve feature's data in the Product Model. They are named "addFeature", "editFeature", "deleteFeature" and "accessFeature".
- Methods to display a 3D geometric representation, called "drawFeature".

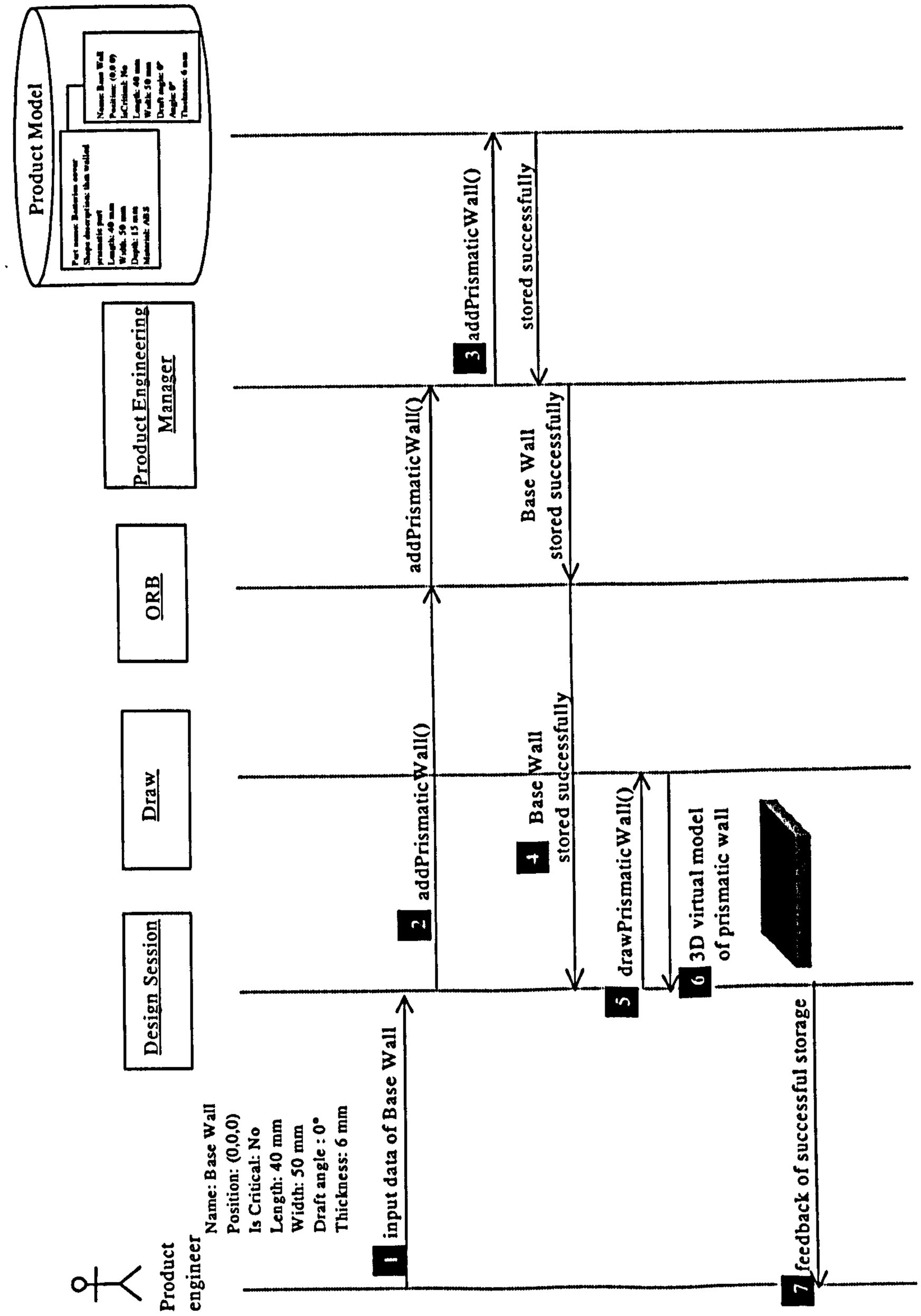


Figure 9.7 The dynamic design model of the "Design Session" class

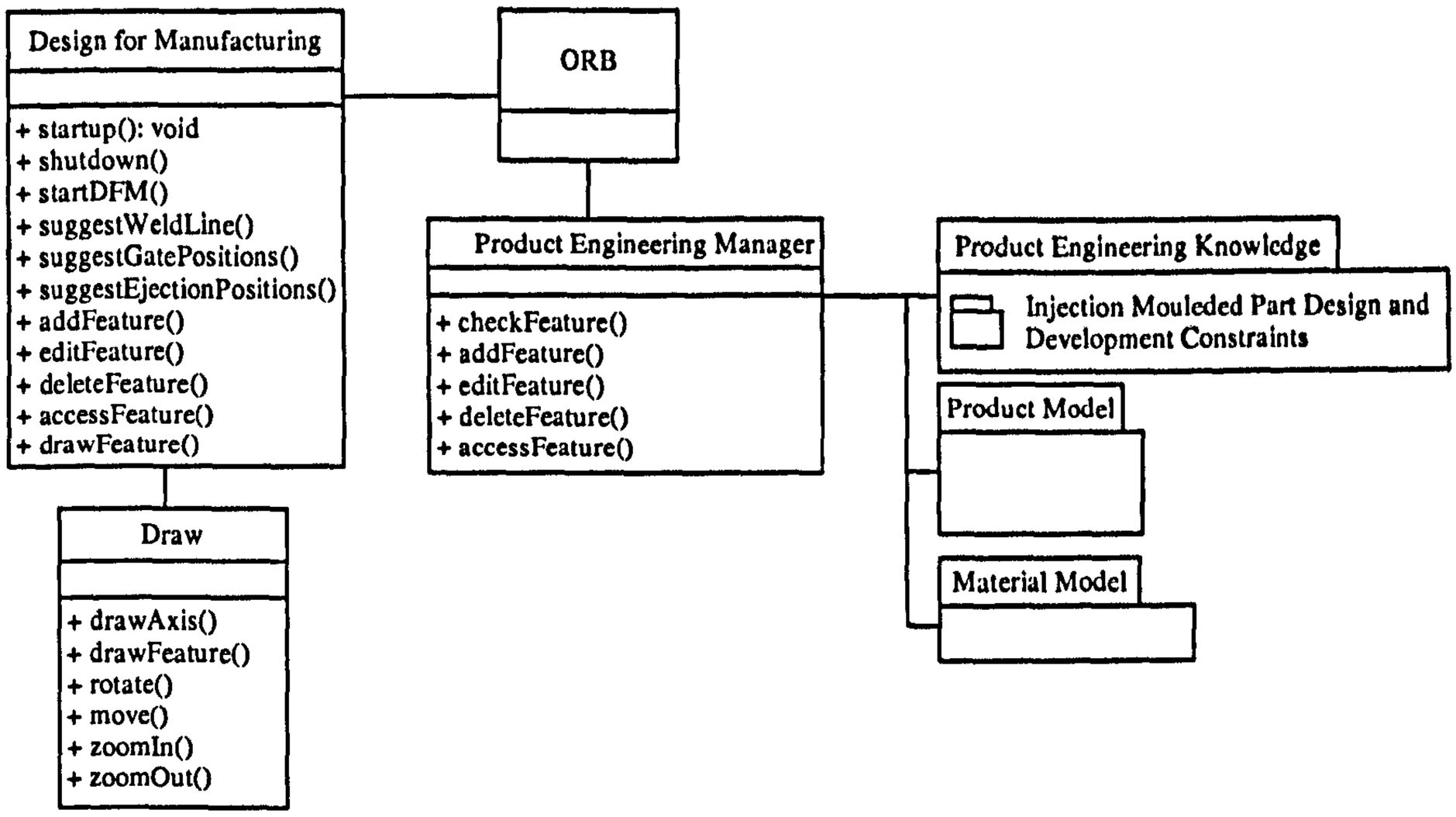


Figure 9.8 The static design model of the "Design for Manufacturing" class

Furthermore, association relationships are represented between the "Design for Manufacturing" class and the "ORB" class as well as the "Draw" class. These relationships are further explained in the dynamic design model.

Dynamic object oriented design model of the "Design for Manufacturing" class

In order to describe the interactions between the objects during the execution of the system, part of the design for manufacturing analysis presented in case study 1 of section 8.2.1 is used. In this scenario, once the product engineer requests the analysis of a plastic part (see figure 9.9-1), the "Design for Manufacturing" object triggers the "startDFM" method and requests this analysis to the "Product Engineering Manager" (see figure 9.9-2). Automatically, the "Product Engineering Manager" object accesses the features definition from the Product Model. For example, the "Base Wall" definition is accessed using the "accessPrismaticWall" method (see figure 9.9-3).

The "Base Wall" definition, which is accessed from the Product Model, is used by the "Product Engineering Manager" object. This object executes the "checkPrismaticWall" method (see figure 9.9-4), which invokes the "Wall Thickness Constraint" object from

the Manufacturing Knowledge Model. Feedback advice is produced and sent back to the "Design for Manufacturing" object (see figure 9.9-5).

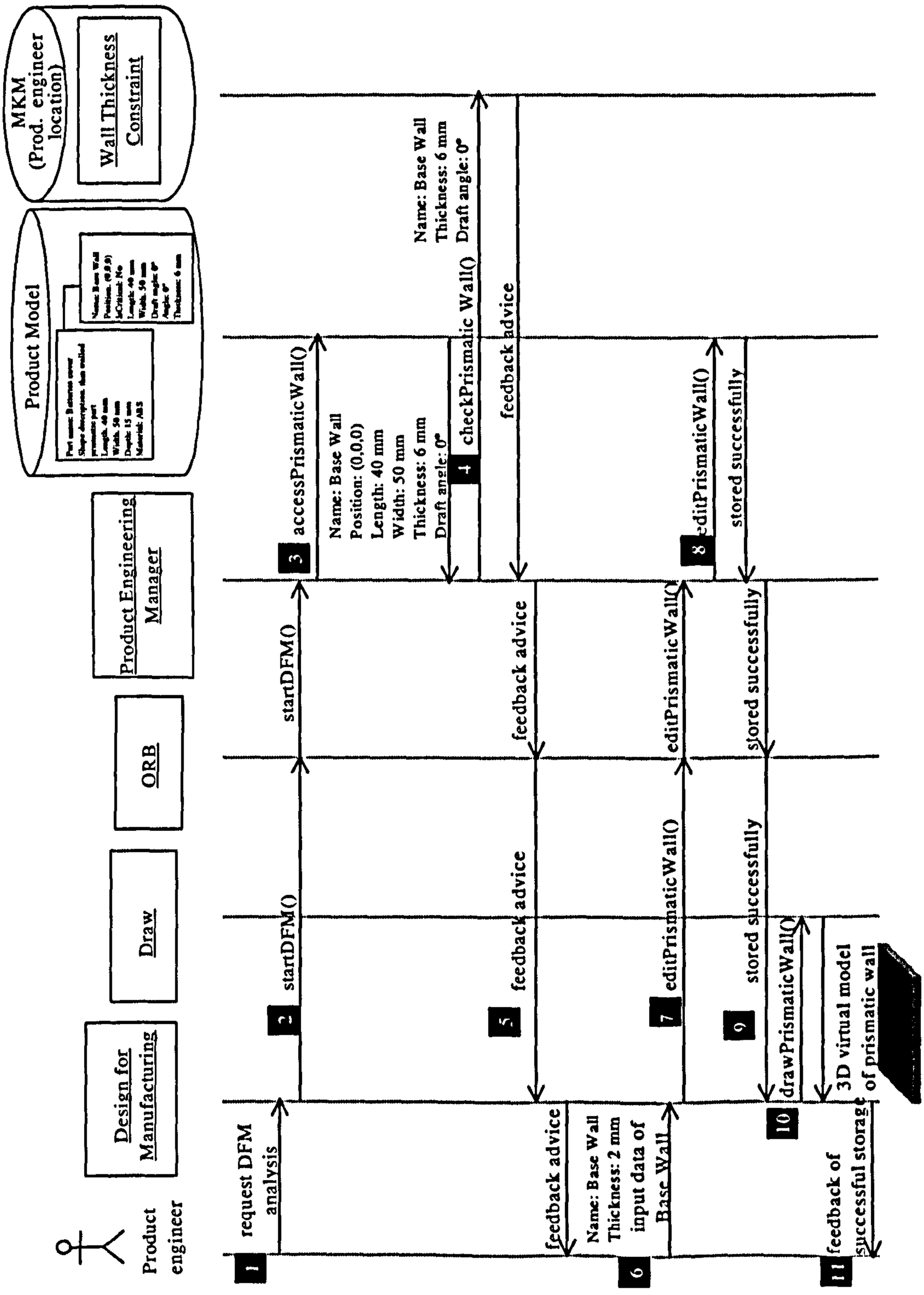


Figure 9.9 The dynamic design model of the "Design for Manufacturing" class

The feedback advice is then provided to the end user, who decides to change the definition of the "Base Wall" by inputting the new definition of the wall (see figure 9.9-

6). This triggers the "editPrismaticWall" method in the "Design for Manufacturing" object (see figure 9.9-7). This method requests the "Product Engineering Manager" object to edit the definition of "Base Wall" in the Product Model. Upon receiving the request, the "Product Engineering Manager" object triggers the "editPrismaticWall" method and stores the new definition of "Base Wall" in the Product Model (see figure 9.9-8).

When successfully storing the new definition, the "Product Engineering Manager" returns a response to the "Design for Manufacturing" object (see figure 9.9-9), which subsequently requests the wall to be redrawn and displays the feedback regarding the successful storage of the feature's definition (see figure 9.9-10,11).

9.4.2.3 The object oriented design model of "Selection of Production Equipment" class

The main functionality of this class, as described in section 6.4.1.4, is to support the selection of the suitable injection moulding machine for the production of a specific plastic part.

Static object oriented design model of the "Selection of Production Equipment" class

In the static design model illustrated in figure 9.10, the "Selection of Production Equipment" class contains the following methods:

- Methods to start and end the object, called "startup" and "shutdown".
- Method to select the suitable injection machine, called "selectMachine"
- Method to calculate the suitable machine characteristics, called "calculateMachineCharacteristics"
- Method to store the suitable injection machine in the Product Model, called "addMachine"

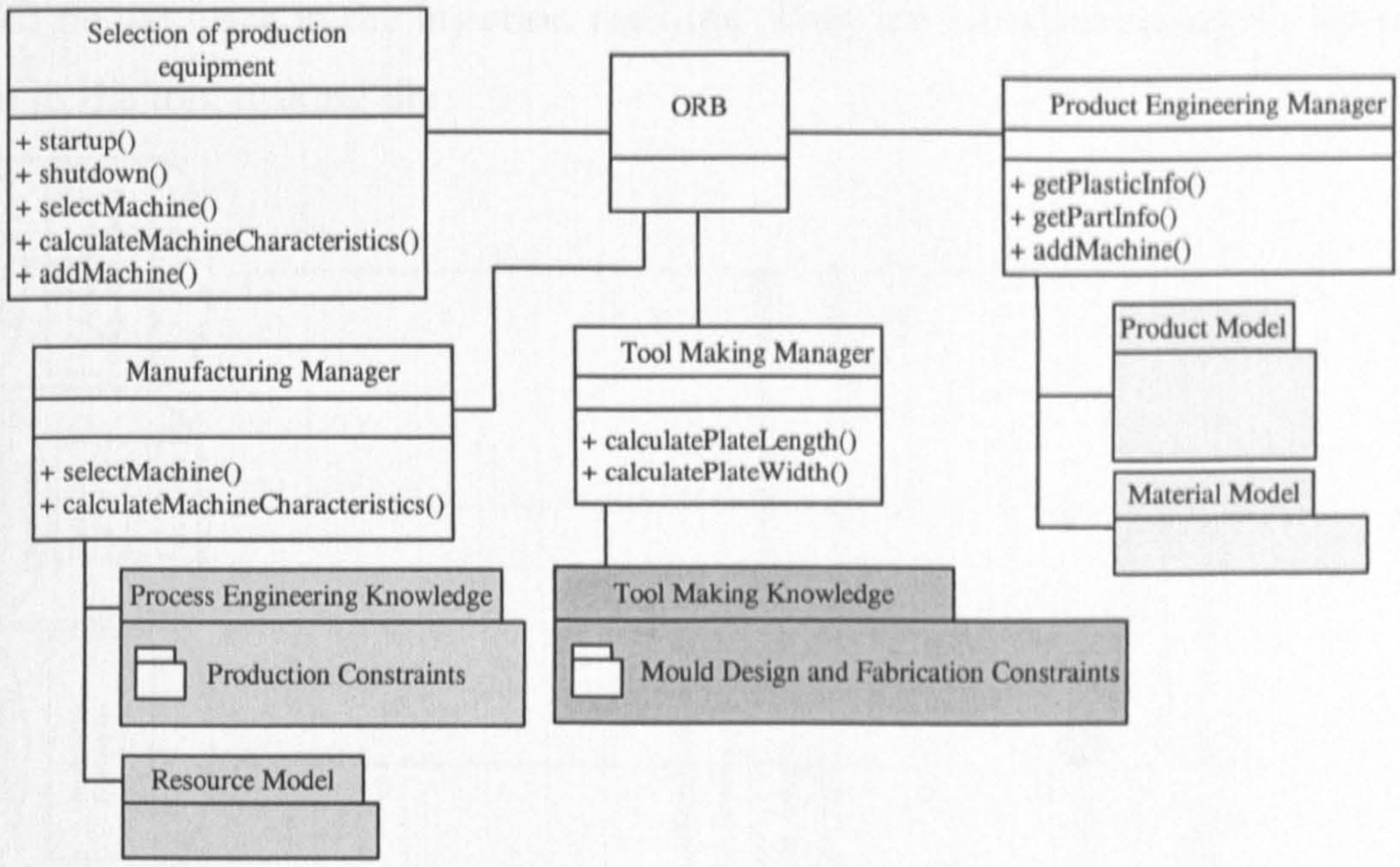


Figure 9.10 The static design model of the "Selection of Production Equipment" class

Dynamic object oriented design model of the "Selection of Production Equipment" class

The scenario presented in case study 4 (see section 8.3.1) is used to illustrate, in the dynamic model of figure 9.11, the interactions between the objects of the system during the selection of a suitable injection machine to produce a plastic part. This dynamic model illustrates how the end user requests from the "Selection of Production Equipment" object, the selection of an injection machine for "Connector cap" plastic part (see figure 9.11-1).

The "Selection of Production Equipment" object then executes the "getPartInfo" and "getMaterialInfo" methods to request the "Product Engineering Manager" to access the definition of "Connector cap" and the data related to the material "Nylon 6" from the Product Model and Material Model (see figure 9.11-2). This data is sent to the "Selection of Production Equipment" object, which then triggers the "calculatePlateLength" and "calculatePlateWidth" method in order to request from the "Tool Making Manager" object, the calculation of suitable mould plates' dimensions (see figure 9.11-3,4). This is because the mould has not been defined yet, as indicated in the product data retrieved form the Product Model. The plates' dimensions are needed for the calculation of the

required tie bar space in the injection machine. They are calculated using the knowledge located in the tool making site.

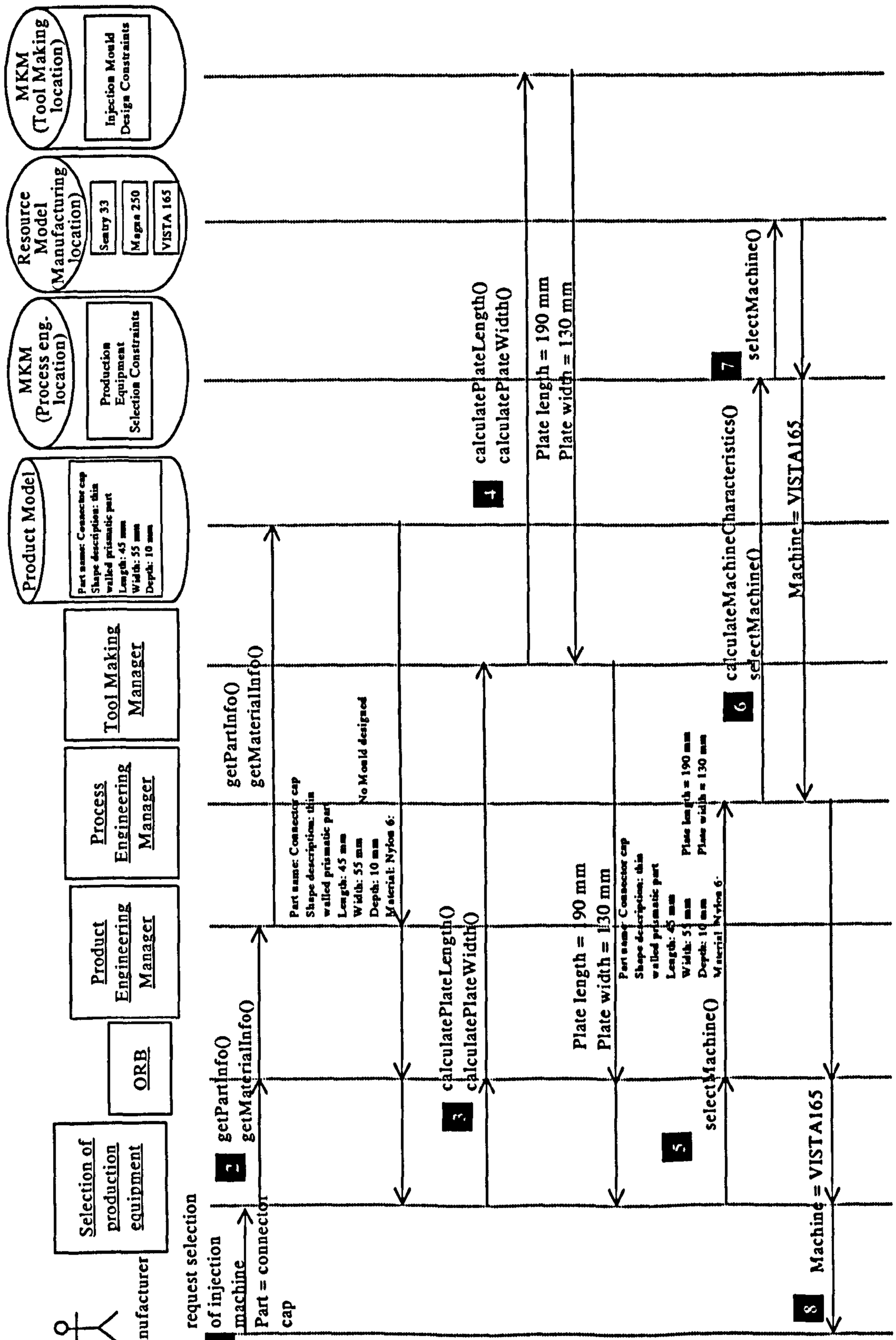


Figure 9.11 The dynamic design model of the "Selection of Production Equipment" class

Once the mould plates' dimensions has been calculated using the "Injection Mould Design Constraints" stored in the Manufacturing Knowledge Model, these are sent back to the "Selection of Production Equipment" object. This object then triggers the "selectMachine" method and sends the data that has been received from the other manager objects to the "Process Engineering Manager" object (see figure 9.11-5). The "calculateMachineCharacteristics" and "selectMachine" methods are executed using the "Production Equipment Selection Constraints" of the Manufacturing Knowledge Model (see figure 9.11-6) and the list of resources stored in the Resource Model (see figure 9.11-7). The produced feedback advice is sent back to the "Selection of Production Equipment" object, which displays the feedback to the end user (see figure 9.11-8).

9.4.2.4 The object oriented design model of "Selection of Process Parameters" class

The main functionality of this class, described in section 6.4.1.4, is to provide advice regarding the optimum operation parameters for the production of a specific plastic part.

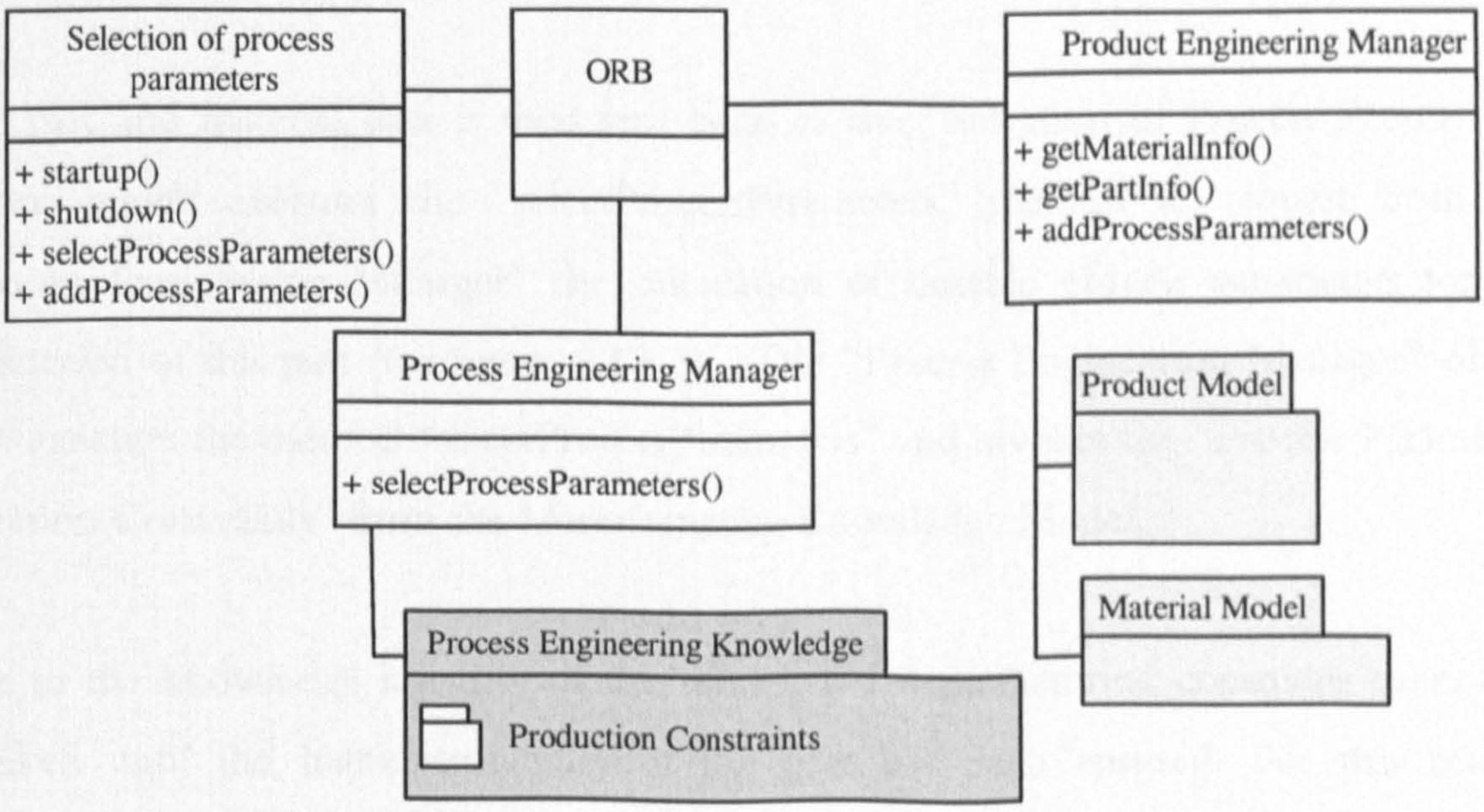


Figure 9.12 The static design model of the "Selection of Process Parameters" class

Static object oriented design model of the "Selection of Process Parameters" class

Figure 9.12 illustrates the static model of the "Selection of Process Parameters" class, which is represented with the following methods:

- Methods to start and end the object, called "startup" and "shutdown"
- Method to select the suitable process parameters, called "selectProcessParameters"
- Methods to store the process parameters in the Product Model, called "addProcessParameters"

Dynamic object oriented design model of the "Selection of Process Parameters" class

The dynamic design model, illustrated in figure 9.13, describes the interactions between the objects of the system during the selection of process parameters for the production of a plastic part, as described in case study 5 (see section 8.4.1).

Once the process engineer selects a plastic part and requests the selection of the suitable process parameters (see figure 9.13-1), the "Selection of Process Parameters" object triggers the methods "getPartInfo" and "getMaterialInfo" (see figure 9.13-2). These methods request from the "Product Engineering Manager" the retrieval of the "Printer's part" and the "ABS" plastic material data from the Product Model and Material Model.

The part and material data is then sent back to the "Selection of Process Parameters" object, which executes the "selectProcessParameters" method to request from the "Process Engineering Manager" the calculation of suitable process parameters for the production of this part (see figure 9.13-3). The "Process Engineering Manager" object then executes the method "selectProcessParameters" and invokes the "Process Parameters Selection Constraints" from the Manufacturing Knowledge Model.

Due to the knowledge integrity of the model, the manufacturing constrains cannot be invoked until the manufacturability of the part has been ensured. For this reason, feedback is sent back to the "Selection of Process Parameters" object (see figure 9.13-4). Thereafter, the object automatically requests, from the "Product Engineering Manager" object, a design for manufacturability analysis of the part definition (see figure 9.13-5).

The "Product Engineering Manager" object triggers the "startDFM" methods and performs the analysis by invoking the "Design Features Manufacturability Constraints" from the Manufacturing Knowledge Model in the product engineering site.

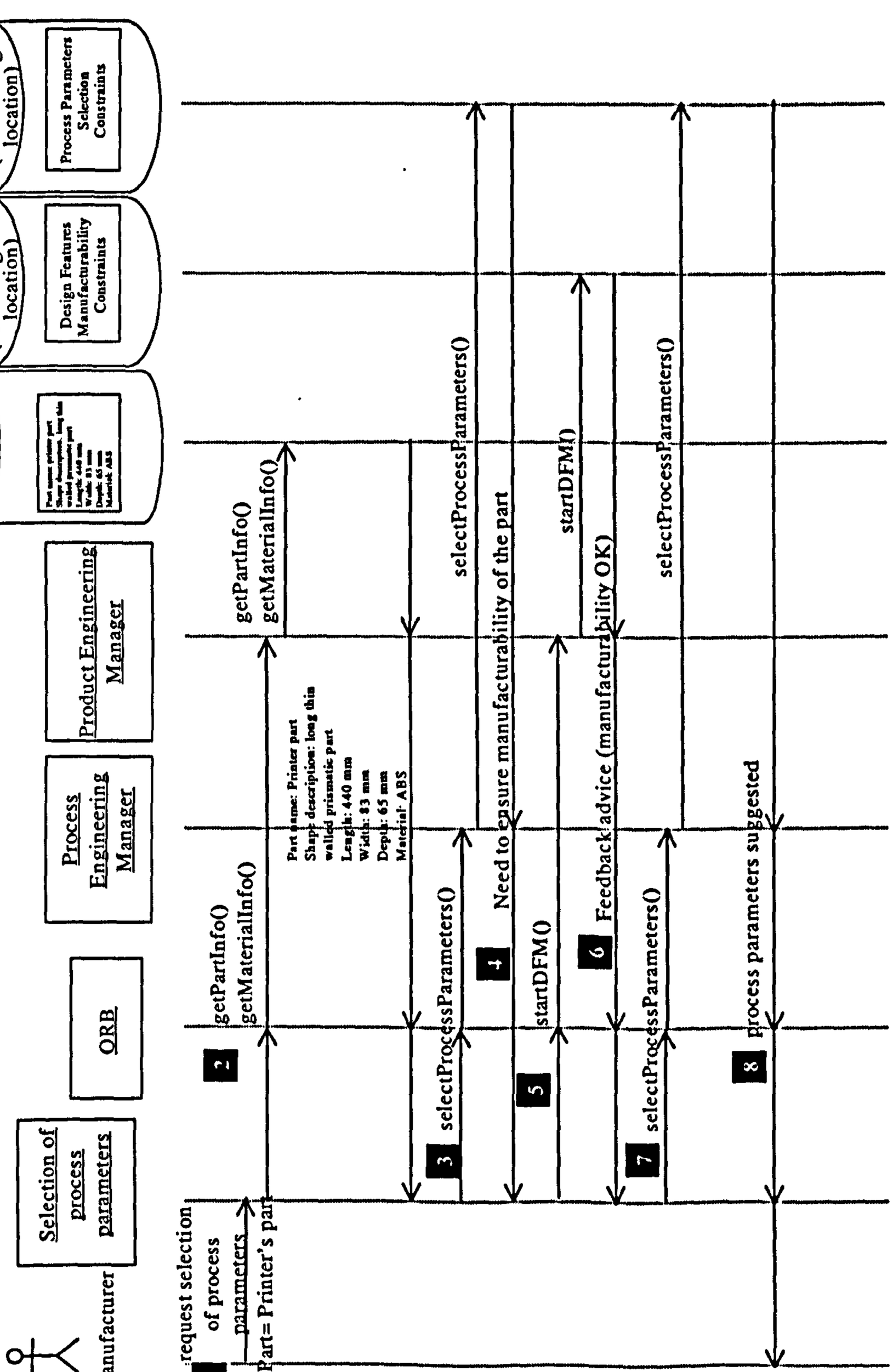


Figure 9.13 The dynamic design model of the "Selection of Process Parameters" class

Thereafter, the "Product Engineering Manager" object sends feedback confirming the part was defined within manufacturing constraints (see figure 9.13-6). Upon receiving this confirmation, the "Selection of Process Parameters" object requests again the calculation of suitable process parameters (see figure 9.13-7). This time, the "Process Engineering Manager" object invokes the "Process Parameters Selection Constraints" from the Manufacturing Knowledge Model and sends back the process parameters suggested (see figure 9.13-8). Finally, the "Selection of Process Parameters" object provides these suggestions to the end user.

9.4.2.5 The object oriented design model of "Mould Design and Fabrication" class

The main functionality of this class is to support the design and fabrication of the different injection mould components (see section 6.4.1.5).

Static object oriented design model of the "Mould Design and Fabrication" class

The "Mould Design and Fabrication" class (see figure 9.14) contains the following methods:

- Methods to start and end the object, called "startup" and "shutdown".
- Methods to suggest the type or dimensions of the mould components. These methods are called "suggestMouldDimensions", and "suggestMouldComponentValues" to simplify its representation. However, the word "MouldComponents" could be replaced by "FeedSystem" or "SprueGate".
- Methods to suggest fabrication techniques, called "suggestFabricationTechniques"
- Methods to store, edit, delete and access components of the mould in the Product Model, called "addMouldComponent", "editMouldComponent", "deleteMouldComponent", "accessMouldComponent".

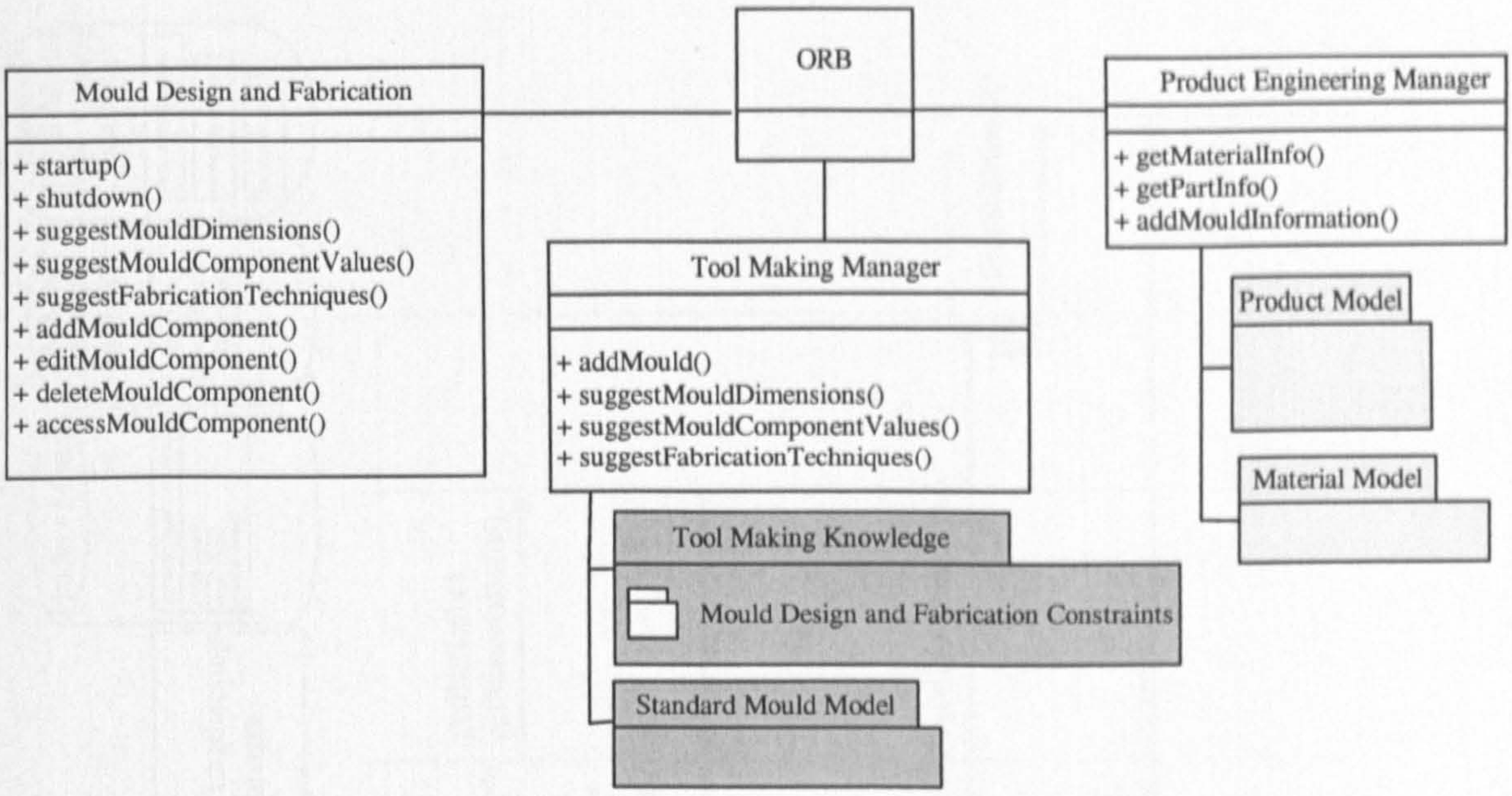


Figure 9.14 The static design model of the "Mould Design and Fabrication" class

Dynamic object oriented design model of the "Mould Design and Fabrication" class

The scenario presented in case study 6 (see section 8.5.1) is used to illustrate the interactions between the objects detailed in the static model of the "Mould Design and Fabrication" class. These interactions are illustrated in the dynamic design model in figure 9.15. In this model, after the toolmaker starts designing the plates of the injection mould for the "Liquid container cap" plastic part, the "Mould Design and Fabrication" object automatically triggers the methods named "getPartInfo" and "getMaterialInfo" (see figure 9.15-1). These methods request from the "Product Engineering Manager" the retrieval of the part and plastic material definition, which is accessed from the Product Model and sent back to the "Mould Design and Fabrication" object. As the machine has not yet been selected, the object requests a number of cavities and their layout to the end user (see figure 9.15-2). This data is required in order to calculate the mould plates length and width.

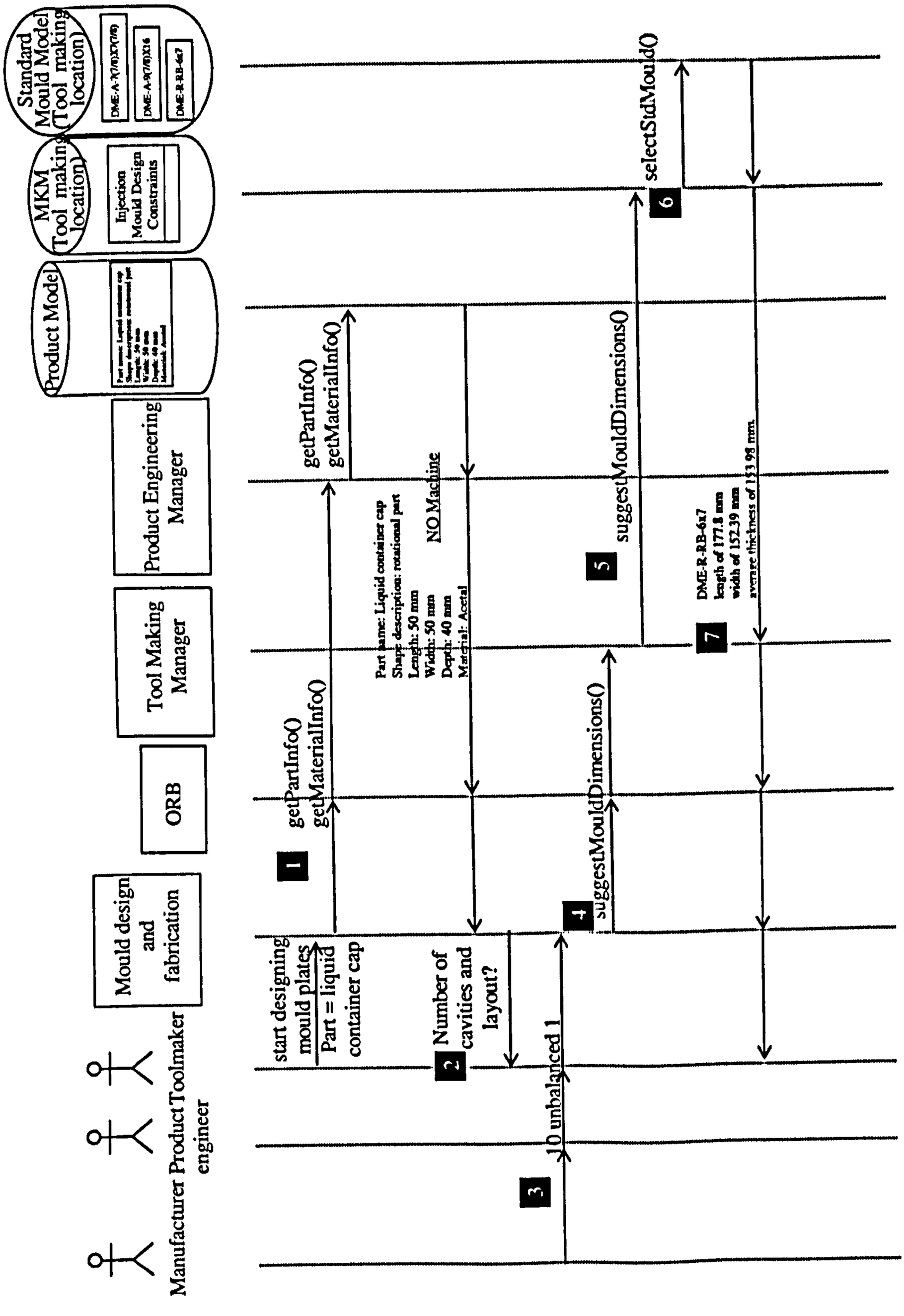


Figure 9.15 The dynamic design model of the "Mould Design and Fabrication" class

The end user(s) then decide(s) to use 10 cavities in an unbalanced arrangement type 1 and to input this data into the "Mould Design and Fabrication" object (see figure 9.15-3). The object triggers the "suggestMouldDimensions" method and request the suggestion of

suitable mould dimensions from the "Tool Making Manager" (see figure 9.15-4). The "Tool Making Manager" then executes the "suggestMouldDimensions" method in order to invoke the "Injection Mould Design Constraints" from the Manufacturing Knowledge Model (see figure 9.15-5). Based on the calculations done by the manufacturing constraints, the method "selectStdMould" is triggered in order to select from the StdMould Model a suitable standard mould that can fit the calculated mould plates (see figure 9.15-6). This data is sent back to the "Tool Maker Manger", which sends the data to the "Mould Design and Fabrication" object. This object provides the data to the end user.

9.5 Closing Remarks

The design of the KdCPD system prototype facilitated its implementation using enabling object oriented technologies. The implementation of such prototype will be presented in the following chapter. This prototype was built in order to demonstrate on of the main innovations of the research, which is to capture, represent and provide product life cycle knowledge to support decision making during collaborative product development.

The object oriented design was done following the guide of CIMOSA reference framework. The use of UML language enabled the reuse of the models that were developed during the information and knowledge modelling presented in chapter 6.

Chapter 10

Object Oriented Implementation of the KdCPD System

10.1 Introduction

This chapter presents the object oriented implementation of a functional KdCPD system prototype, which design was presented in the previous chapter. The prototype was developed as a web based system, as this was the most suitable enabling technology to implement such a collaborative environment. The use of this technology provided the following advantages:

- The system is platform independent (hardware and operative system)
- The application's code is portable and independent of the web browser used by the engineer.
- The system can be accessed regardless the time and location of the data, knowledge and engineers who access the system.
- The update and maintenance of the software is facilitated as it resides in a server instead of in individual user's computer.

The following sections present the implementation of the different layers of the KdCPD system prototype.

10.2 Information layer implementation

As explained in section 9.4.1, the information layer contains the Manufacturing Knowledge Model, Product Model, Organisation Model, Engineering Data Models, database managers and the ORB. Their implementation is described in the following subsections.

10.2.1 Implementation of the Manufacturing Knowledge Model

The Manufacturing Knowledge Model was implemented as a database using the ObjectStore™ object oriented database management system (OODBMS) version 6.0 (Progress Software 2003). ObjectStore can run on Windows or Unix environment and has support for Java and C++ languages. This OODBMS was selected because its object oriented approach can effectively implement and manage the complexity of the model.

The ObjectStore database management system was running on a Silicon Graphics (Silicon Graphics 2003) workstation using Windows NT operative system. In order to simulate the geographically distributed location of the knowledge, the workstation was functioning as a server containing different database schemas in separate logical servers. A logical server simulates an independent computer server running a database simultaneously to other databases, which are running in other independent servers. Hence, three different logical server were implemented to contain the knowledge located in the product engineering, process engineering and tool making company. For the implementation of the model in the OODBMS, Java™ (Sun Microsystems 2003) enabling technology was selected as this language is also suitable for implementing the web based engineering applications, which access the product life cycle data and knowledge from the databases.

The Manufacturing Knowledge Model represented in UML object oriented language (see chapter 7) provided a blueprint for the implementation of the model in the OODBMS. The following paragraphs describe in more detail the implementation of the UML notation in Java language.

Implementation of the manufacturing constraint classes using Java

Each of the classes of the Manufacturing Knowledge Model was implemented as a Java class using the same name as defined using the UML notation. The manufacturing constraint classes inherit from the "Attribute Constraint" class, as defined in section 9.4.1.1. For example, figure 10.1 illustrate the implementation of the class called "Vent Positions Constraint", which was described in section 7.4.4.4. This Java class inherits

from the "Attribute Constraint" class and contains a list of relationships with the constraints: "Gating System Design Constraint" (see figure 10.1-a), "Reinforcement Draft Angle Constraint", "Reinforcement Base Radius Constraint" and "Reinforcement Distance Constraint" (see figure 10.1-b). These relationships are added with the method "super.insertBackwardRelationship", which was implemented for the "AtrributeConstraint" class.

```
public class VentPositionsConstraint extends AttributeConstraint
{
    //CONSTRUCTOR
    public VentPositionsConstraint(String name, String description)
    {
        super(name, description);

        super.InsertBackwardRelationship("GatingSystemDesignConstraints","GatingSystem",true,"ToolMaker");   (a)

        super.InsertBackwardRelationship("ReinforcementDraftAngleConstraint","Reinforcement",false,"ProductEngineer");   (b)
        super.InsertBackwardRelationship("ReinforcementBaseRadiusConstraint","Reinforcement",false,"ProductEngineer");
        super.InsertBackwardRelationship("ReinforcementDistanceConstraint","Reinforcement",false,"ProductEngineer");
}
```

Figure 10.1 Implementation of the "Vent Positions Constraint" class in Java language

Implementation of the methods in a manufacturing constraint using Java

The recommendation and limitation rules, which represent the manufacturing constraints in the Manufacturing Knowledge Model, were implemented as methods of a Java class. For example, figure 10.2 illustrates the implementation of the "Wall Thickness Constraint" class, as defined in section 7.4.1.1. The recommendation and limitation rule of this constraint is implemented in the "suggestValue" and "checkValue" method of the class (see figure 10.2-a,b).

Implementation of the generalisation and aggregation relationships using Java

The relationships among the Manufacturing Knowledge Model's classes were implemented as follows:

- Generalisation relationship: This relationship was implemented by including in the code after the name of the inherited class, the "extends" keyword, followed by the name of the parent class as follows:

  class RibConstraints *extends* ReinforcementConstraints

```
public class WallThicknessConstraint extends AttributeConstraint
{
    //METHODS
    public String suggestValue(Plastic plastic, float wall_thickness, boolean first_wall, float thickness_first_wall){
        String result = "";
        float minwallthickness = plastic.getMinWallThickness();
        float maxwallthickness = plastic.getMaxWallThickness();
        if ((!first_wall)and(wall_thickness >= minwallthickness) and (wall_thickness <= maxwallthickness)) result = result + "" The
thickness should be the same of the first wall: "+thickness_first_wall+" mm.\n";
        else result = result + "" The thickness should be between "+minwallthickness+" mm. and "+maxwallthickness+" mm.
Recommended: "+ (minwallthickness+maxwallthickness)/2)+" mm.\n";
        if (wall_thickness>=maxwallthickness) result = result + "" A rib can be used for stiffness.\n";
        return result;
    }

    public boolean checkvalue(float wall_thickness, Plastic plastic, boolean first_wall, float thickness_first_wall){
        float minwallthickness = plastic.getMinWallThickness();
        float maxwallthickness = plastic.getMaxWallThickness();
        if ((!first_wall)and(wall_thickness >= minwallthickness) and (wall_thickness <= maxwallthickness)) return (wall_thickness ==
thickness_first_wall);
        else return (( wall_thickness >= minwallthickness) and (wall_thickness <= maxwallthickness));
    }
```

Figure 10.2 Implementation of the methods of the "Wall Thickness Constraint" class

- Aggregation relationship: an example of the implementation of such relationship is shown in figure 10.3. The code illustrated in this figure implements the aggregation relationship for the "Wall Constraints" class, which was shown in figure 7.11. In this class, the values of the attributes are the instances of the following classes: "Wall Thickness Constraint", "Wall Transition Constraint" and "Wall Draft Angle Constraint".

```
public class WallConstraints
{
    //ATTRIBUTES
    private WallThicknessConstraint wall_thickness_constraint;
    private WallTransitionConstraint wall_transition_constraint;
    private WallDraftAngleConstraint wall_draft_angle_constraint;

}
```

Figure 10.3 Implementation of the aggregation relationship for the "Wall Constraints" class

Implementation of the packages using Java

The packages in the Manufacturing Knowledge Model were implemented as folders in the Windows environment. For example, the manufacturing constraints that belong to the "Design Features Manufacturability Constraints" package, shown in figure 7.49, were implemented as Java classes and placed under a folder named DFMC, which is an abbreviation for "Design Features Manufacturability Constraints".

As a requirement of the Object Store OODBMS, the implemented classes were then made persistence capable. This means that upon termination of the application, the state of the objects are saved and restored when the system is restarted. The classes are made persistence capable by running a postprocessor, which adds coding into the classes to provide the new required functionality.

Finally, after the classes and folders had been implemented, these were placed inside the corresponding partner's site logical server. The following subsections present in more detail the folder structure of each implemented logical server: product engineering, process engineering and tool making logical server.

10.2.1.1 Folder structure in the product engineering logical server

Figure 10.4 illustrates the product engineering logical server. In this server, a folder called "Design Features Manufacturability Constraints" (DFMC) contains folders with the Java classes that implement the manufacturability constraint of the design features, as described in section 7.4.1.9. An example is the "Wall Constraints" folder, which structure was described in section 7.4.1.1. This folder contains the following classes: "Wall Constraints", "Wall Thickness Constraint", "Wall Transition Constraint" and "Wall Draft Angle Constraint".

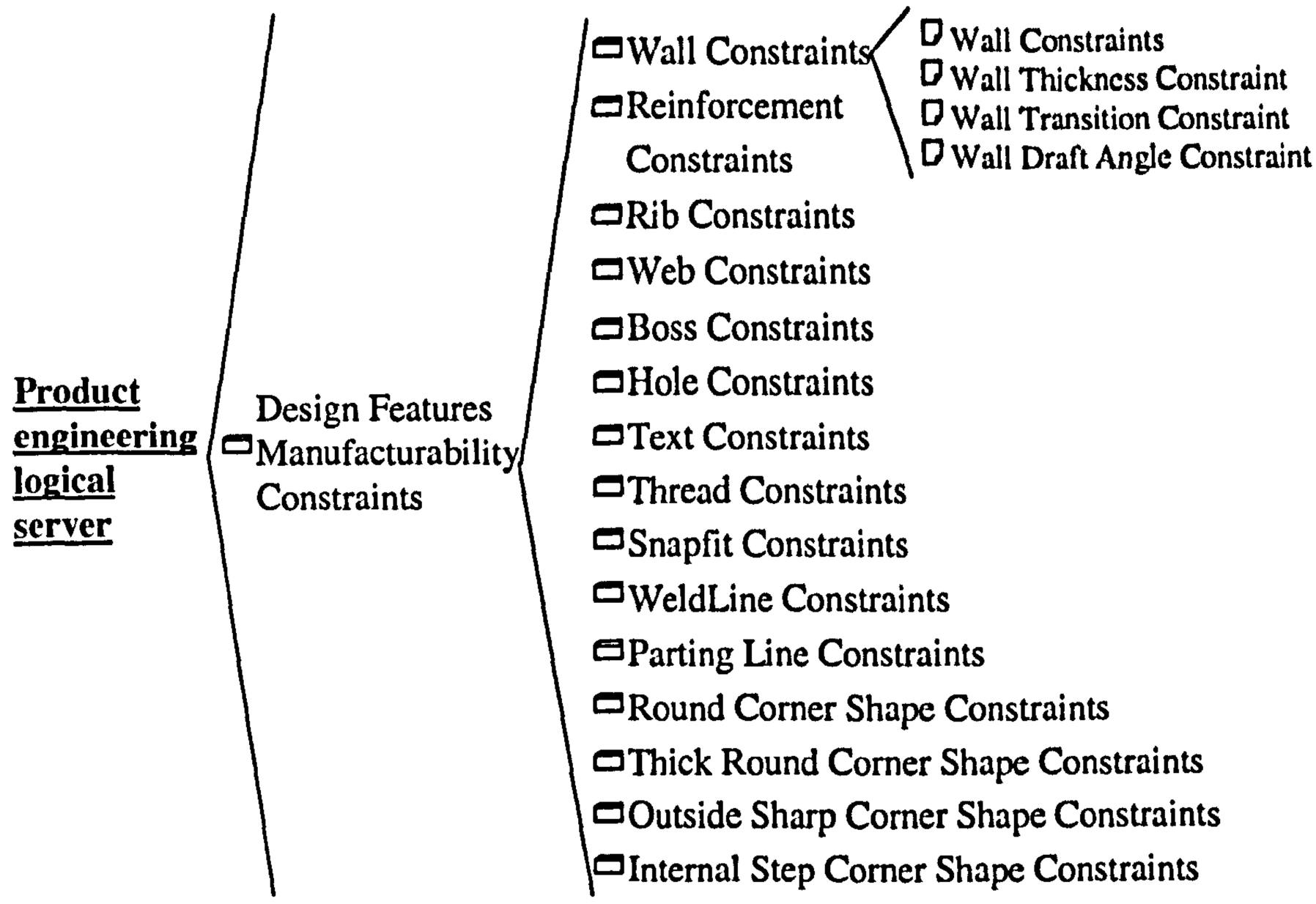


Figure 10.4 The folder structure of the Manufacturing Knowledge Model located in the product engineering logical server

10.2.1.2 Folder structure in the process engineering logical server

The folder structure of the process engineering logical server is illustrated in figure 10.5. This server contains a folder called "Production Constraints" (PC), which contains two folders called "Process Parameters Selection Constraints" and "Production Equipment Selection Constraints". These folders contain the Java classes that implement the manufacturing constraints to support the selection of production equipment and process parameters as described in section 7.4.2 and 7.4.3.

10.2.1.3 Folder structure in the tool making logical server

The tool making logical server contains two folders called "Injection Mould Design Constraints" and "Injection Mould Fabrication Constraints" (see figure 10.6). These folders contain the manufacturing constraints that limit the design and fabrication of an injection mould as explained in section 7.4.4 and 7.4.5.

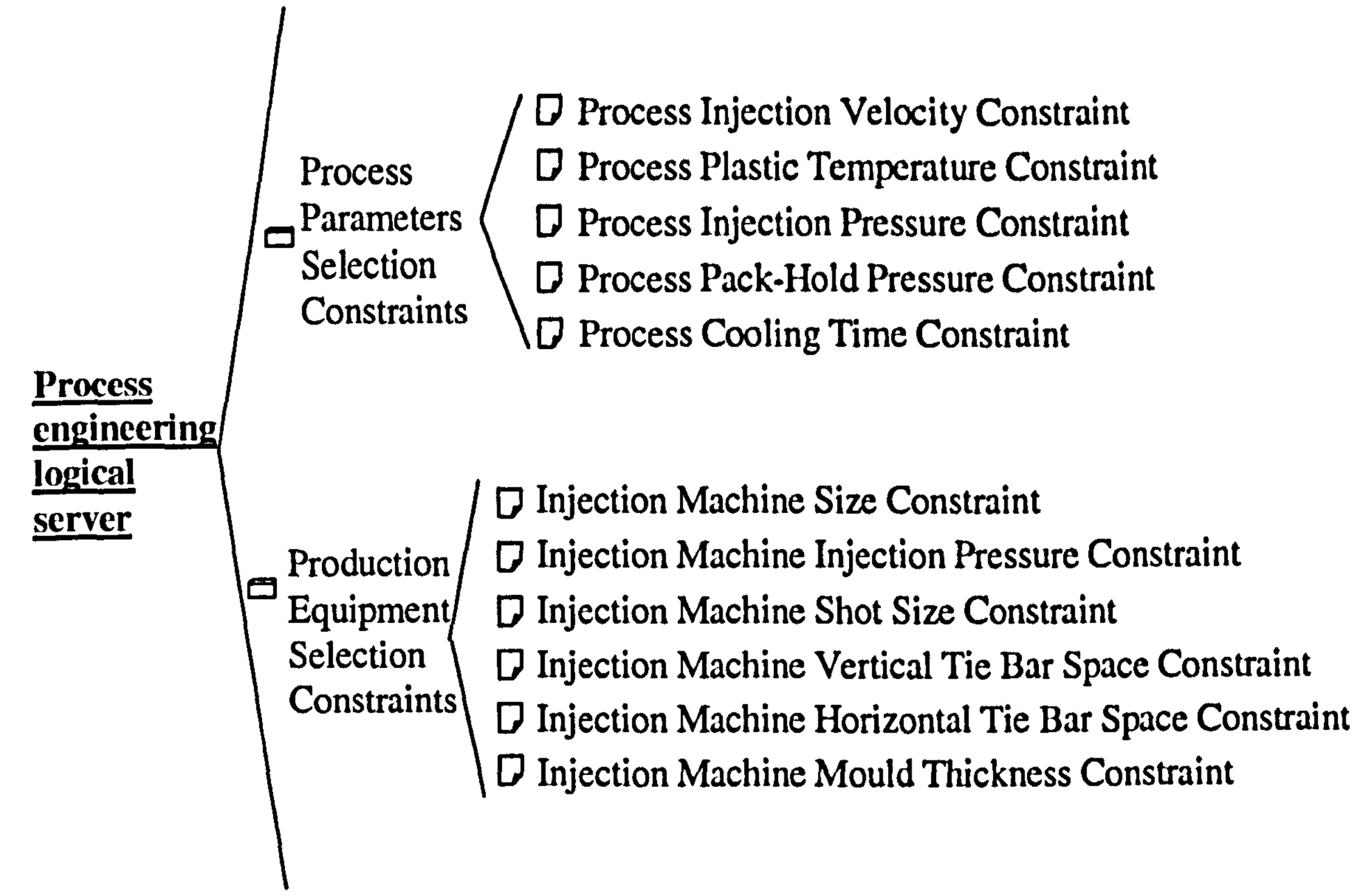


Figure 10.5 The folder structure of the Manufacturing Knowledge Model located in the process engineering logical server

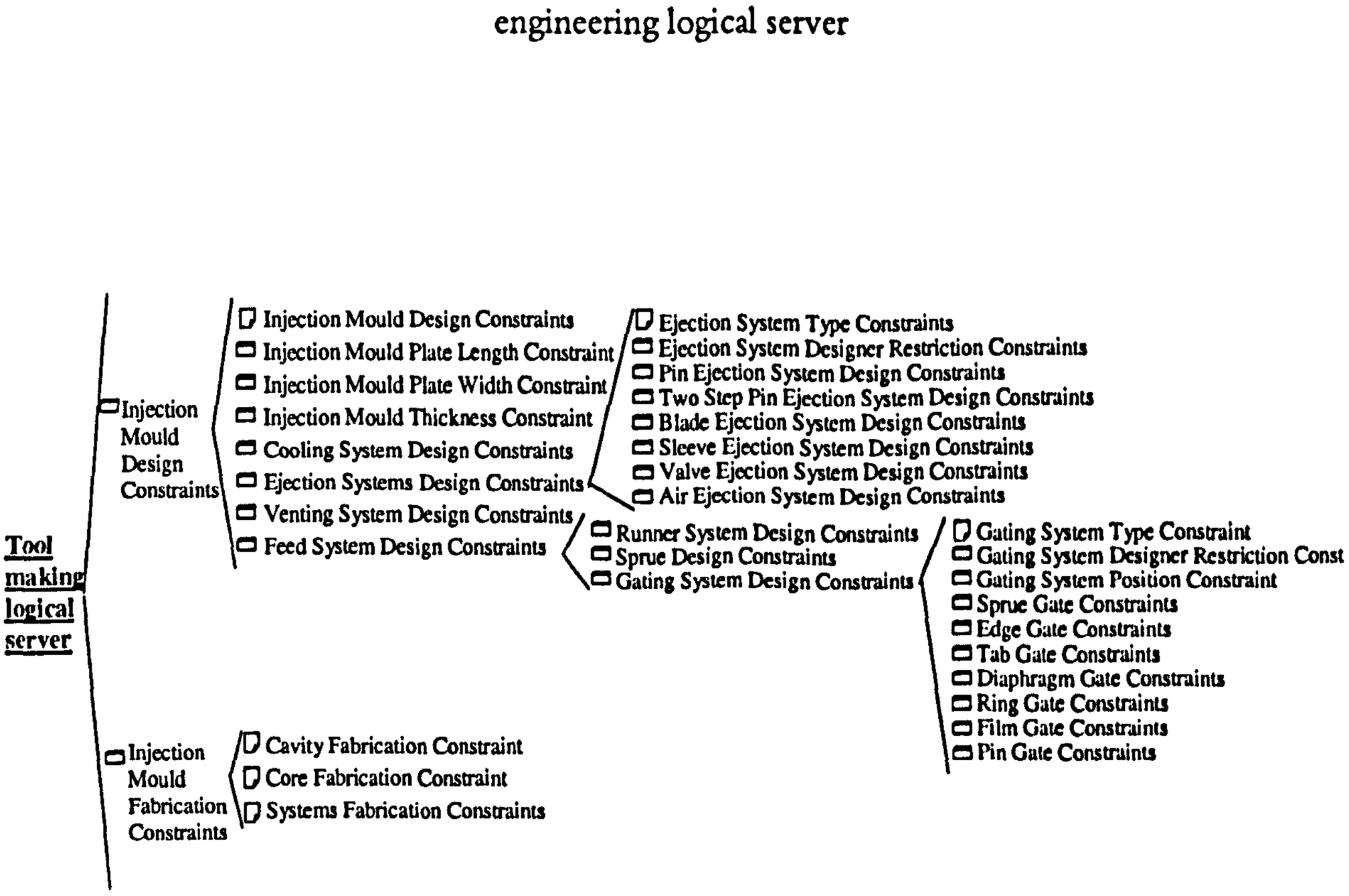


Figure 10.6 The folder structure of the Manufacturing Knowledge Model located in the tool making logical server

10.2.2 Implementation of the Product Model, Organisation Model and Engineering Data Models

The Product Model, Organisation Model and Engineering Data Models represented in UML notation in chapter 7 provided a blueprint for the implementation of these models in the Object Store™ OODBMS. This database management system was used as it facilitated the management of geographically distributed product data and team members' information. The models were implemented following the same approach as the one used for the Manufacturing Knowledge Model.

10.2.3 Implementation of the database managers

The database managers were implemented in order to manage the storage and retrieval of knowledge and data from the databases. As explained in section 9.4.1.2, these were implemented within the logical server of each partner. In order to provide the main functionality of the manager class, which is to gain access to the databases and store/modify data, the following functionality was implemented:

- Opening and closing a database
- Managing transactions: Transactions are required in order to have data integrity in the database. As such, any changes made to the database during the transaction are not immediately incorporated into the database. This is only updated after the transaction has been finalised and no problems have been detected. If the transaction fails, the database is not updated (Sommerville 2001).
- Managing sessions: In the ObjectStore environment, a session is required in order to open a database and execute a transaction.
- Generating an entry point: In order to access objects in a database, it is required to have an entry point as a mechanism for referring to these objects. In ObjectStore, the entry points are called "roots". Once one object is retrieved through this root, any object that relates to it can be retrieved by navigating object references. Each database typically has a relatively small number of entry point objects, each of which allows access to a large collection of related objects. The database root needs to be created inside a transaction and its name needs to be unique.

- Adding, updating and removing database objects: This always takes place within a transaction.

10.2.4 Implementation of the ORB middleware

The implementation of a system architecture, such as the KdCPD system, which has a distributed object oriented client-server approach requires of middleware (object request broker) in order to handle the communication between the distributed objects. In the KdCPD system, the distributed objects are the decision support engineering applications, the information and knowledge databases and the managers. Their communication is handled through the object request broker: CORBA (Object Management Group 2003).

CORBA (Common Object Request Broker Architecture) is the standard distributed object architecture developed by the Object Management Group (OMG) consortium. This standard architecture allows CORBA objects to invoke other objects across networks and operating systems without knowing where they reside or in what language they are implemented. For this, the language independent Interface Definition Language (IDL) is used to define the interfaces between the objects (Sun Microsystems 2003). Furthermore, CORBA objects have a unique identifier called an Interoperable Object Reference (IOR). This IOR is used when one object requests services from another (Sommerville 2001). Hence, if an object wishes to use services provided by another object then it accesses these services through the IDL interface using the IOR. For example, the "Design for Manufacturing" object requests from the "Product Engineering Manager", through the IDL interface, the service of storing the part definition (see figure 10.7).

The CORBA standard has been implemented by different vendors. The Java IDL programming model by Sun Microsystems (2003) was selected for this research. This is because the Java IDL enables the decision support engineering applications to transparently invoke operations from the geographically distributed database managers.

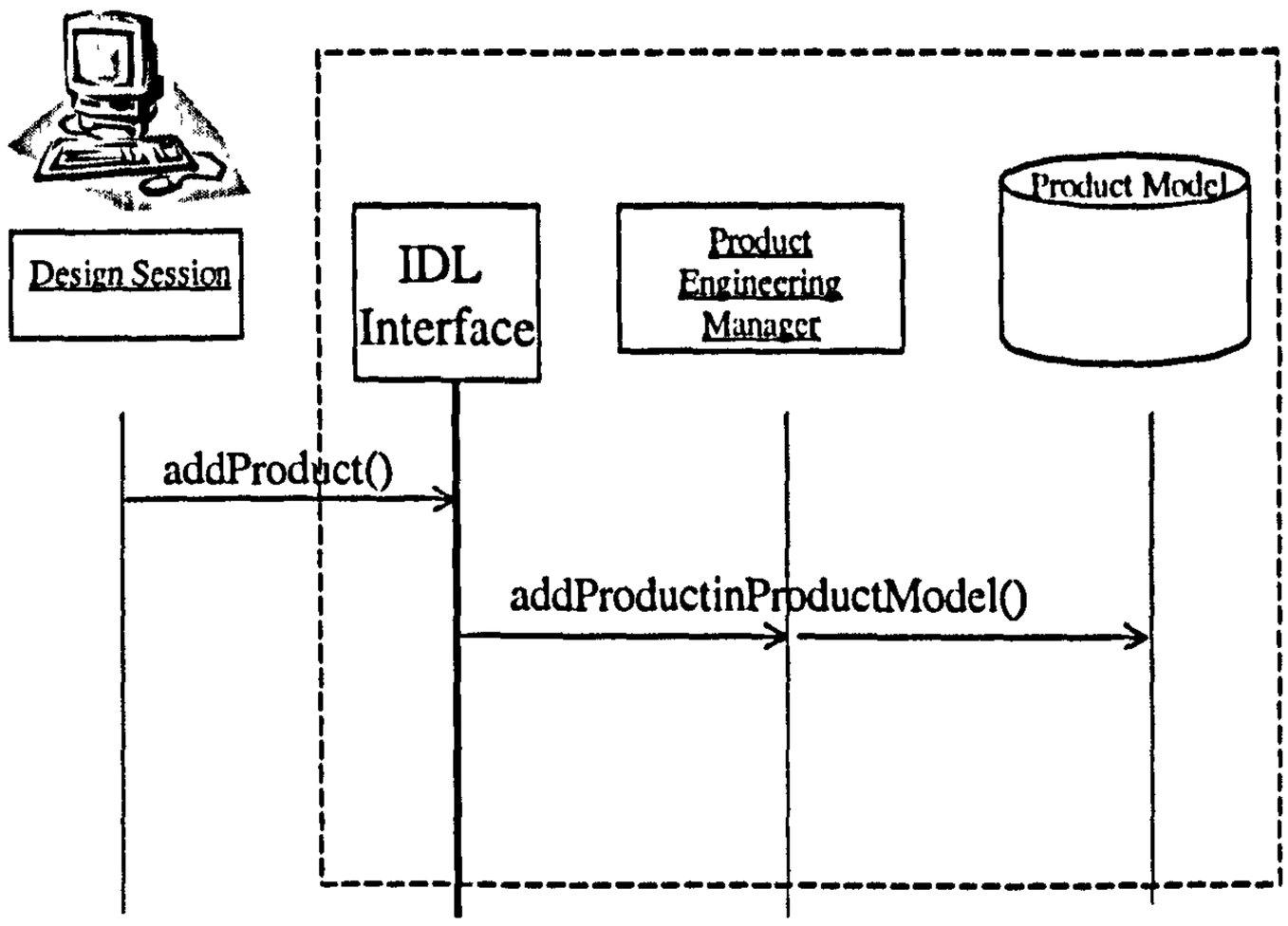


Figure 10.7 IDL interface for access services between objects in the distributed object oriented system architecture

For the implementation of the KdCPD system prototype, it was required to implement middleware for each logical server (product engineer, process engineer and toolmaker). The procedure followed to implement the middleware for the three logical servers is presented next:

1. Firstly, the methods of the managers were implemented in the IDL interface (see figure 10.8-1).
2. As shown in figure 10.8-2, this interface was then compiled. The compiler provided by Java IDL generates the java version of the interface, as well as the class code files for the skeleton. The purpose of this skeleton is to translate a remote call and any parameters to their implementation specific format. In addition, it calls the method that needs to be invoked. When the method returns, the skeleton code transforms results or errors, and sends them back to the client via the ORB.
3. Once the skeleton was generated, this was used to implement a server application for accessing the database manager. The server application implemented the methods included in the interface and calls its respective method from the database manager (see figure 10.8-3). In addition the server code includes a mechanism to start the ORB and wait for invocation from a remote client.

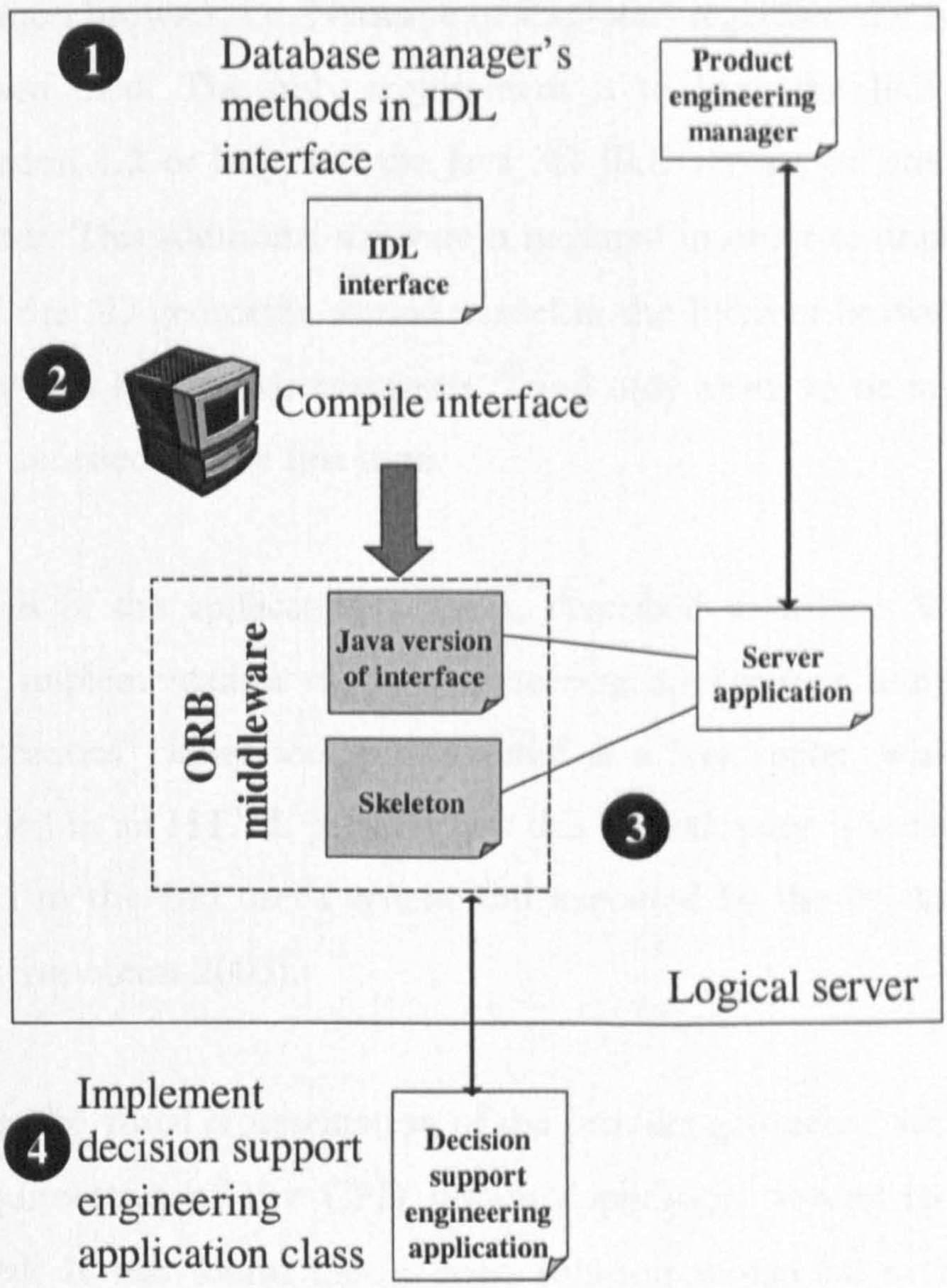


Figure 10.8 Implementation of the ORB middleware

4. The decision support engineering applications were then implemented, as illustrated in figure 10.8-4. The implementation code in the applications searches for the middleware, obtains a reference for the remote object, and calls its method.
5. Finally, the middleware service is started. This service runs continuously in the logical server and its purpose is to be idle waiting for requests to the middleware.

10.2 Implementation of the application layer

The engineering applications were implemented with object oriented enabling technologies, such as Java™ (Sun Microsystems 2003) and Java™ 3D languages for the geometry display. These languages were used because the applications can be accessed

through any common browser, i.e. Netscape or Explorer, regardless the type of computer or operative system used. The only requirement is to have the JRE (Java Runtime Environment) version 1.2 or later and the Java 3D JRE version or later installed in the end user's computer. This additional software is required in order to display the graphical user interface and the 3D geometric virtual model in the Internet browser. The software is provided free of cost by Sun Microsystems™ and only needs to be installed when the KdCPD system is accessed for the first time.

The design models of the applications' classes, described in section 9.4.2, provided a blueprint for the implementation of the engineering applications using Java language. Each of the applications' classes was implemented as a Java applet, which is a program that can be included in an HTML page. When this HTML page is accessed, the applet's code is transferred to the end user's system and executed by the browser's Java Virtual Machine (Sun Microsystems 2003).

In order to address the visual representation of the product geometry, which is one of the technological requirements of any CPD system application, several technologies were initially considered. It was found that a good solution would be to integrate a solid modelling tool, such as the provided by the commercial 3D environments; i.e., CoCreate (2003), Dassault Systems (2003); Informative Graphics Corp. (2003); PTC (2003). However, the objectives of the object oriented design and implementation of the KdCPD system prototype was to demonstrate how product life cycle knowledge could support decision making during CPD. Thus, the research effort was focused in achieving such a knowledge based system and its integration with a commercial solid modelling tool is an issue beyond the scope of this research and remains to be addressed in further work. Nevertheless, the author experimented with Java 3D technology in order to develop a 3D environment, where a virtual geometric model was built based on product data captured in the Product Model. Java 3D was used to implement the automatic generation of a feature's geometric model in real time. The drawback of this enabling technology is that in order to generate a complex geometry equal to the one provided by

the solid modelling tools many man hours are required to build a complete library of geometries.

The following subsections present the implementation of the graphical user interface of the engineering applications, which were implemented following the design models described in section 9.4.2.

10.2.1 Graphical user interface of the Design Session application

This application was implemented as a Java applet following the design models described in section 9.4.2.1. Figure 10.9 shows the graphical user interface of this applet, which contains five main areas: operation menu, feedback area, menu of mouldable features, input area and geometric representation area.

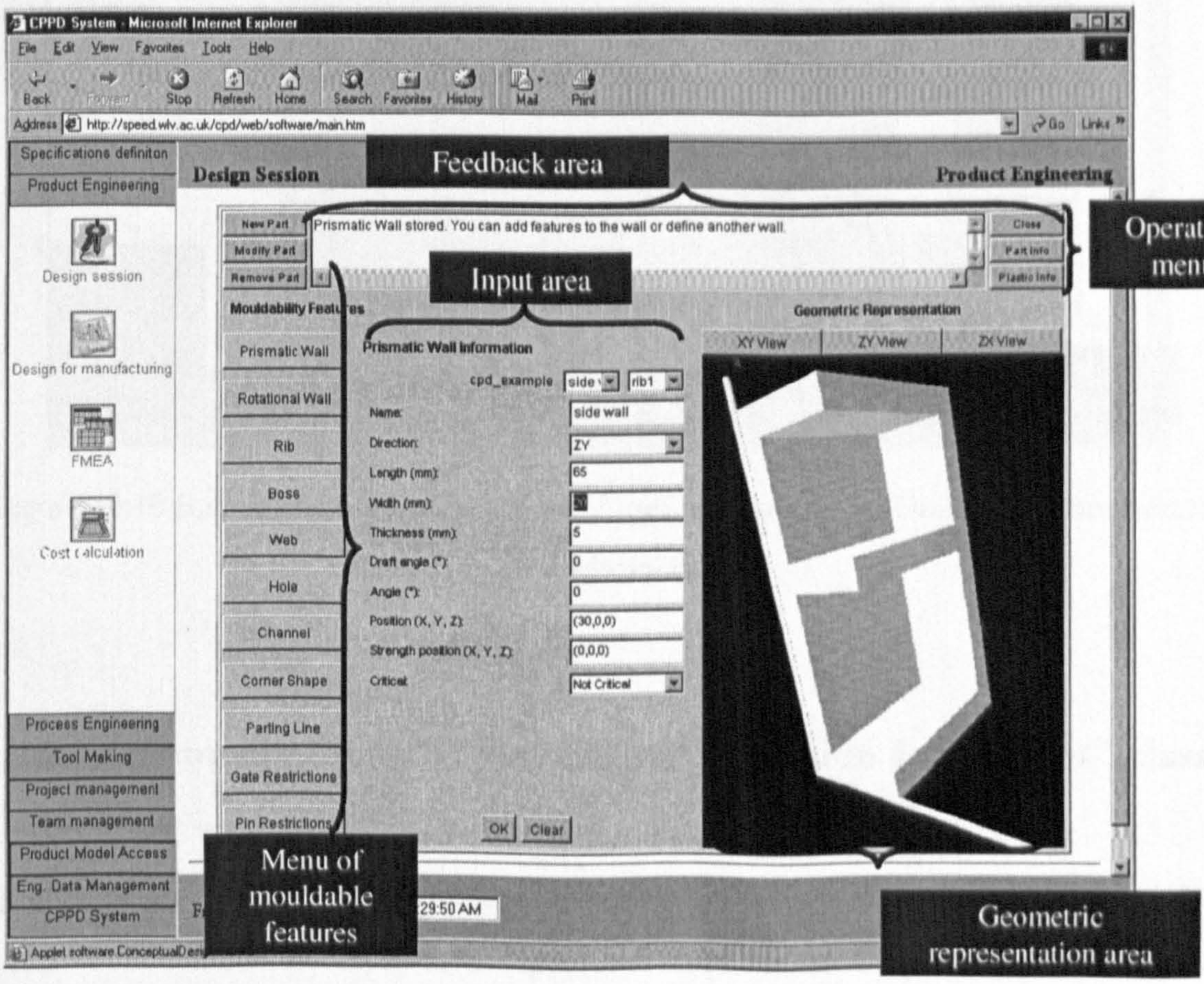


Figure 10.9 Implementation of the graphical user interface of the Design Session class

10.2.2 Implementation of "Design for Manufacturing" class

Figure 10.10 shows the graphical user interface of the "Design for Manufacturing" application, which was implemented following the design models of section 9.4.2.2. The interface is very similar to the one of the "Design session" application. The main difference is a "Start DFM (Design For Manufacturing)" button in the operation menu. This button was included in order to provide the main functionality of the class.

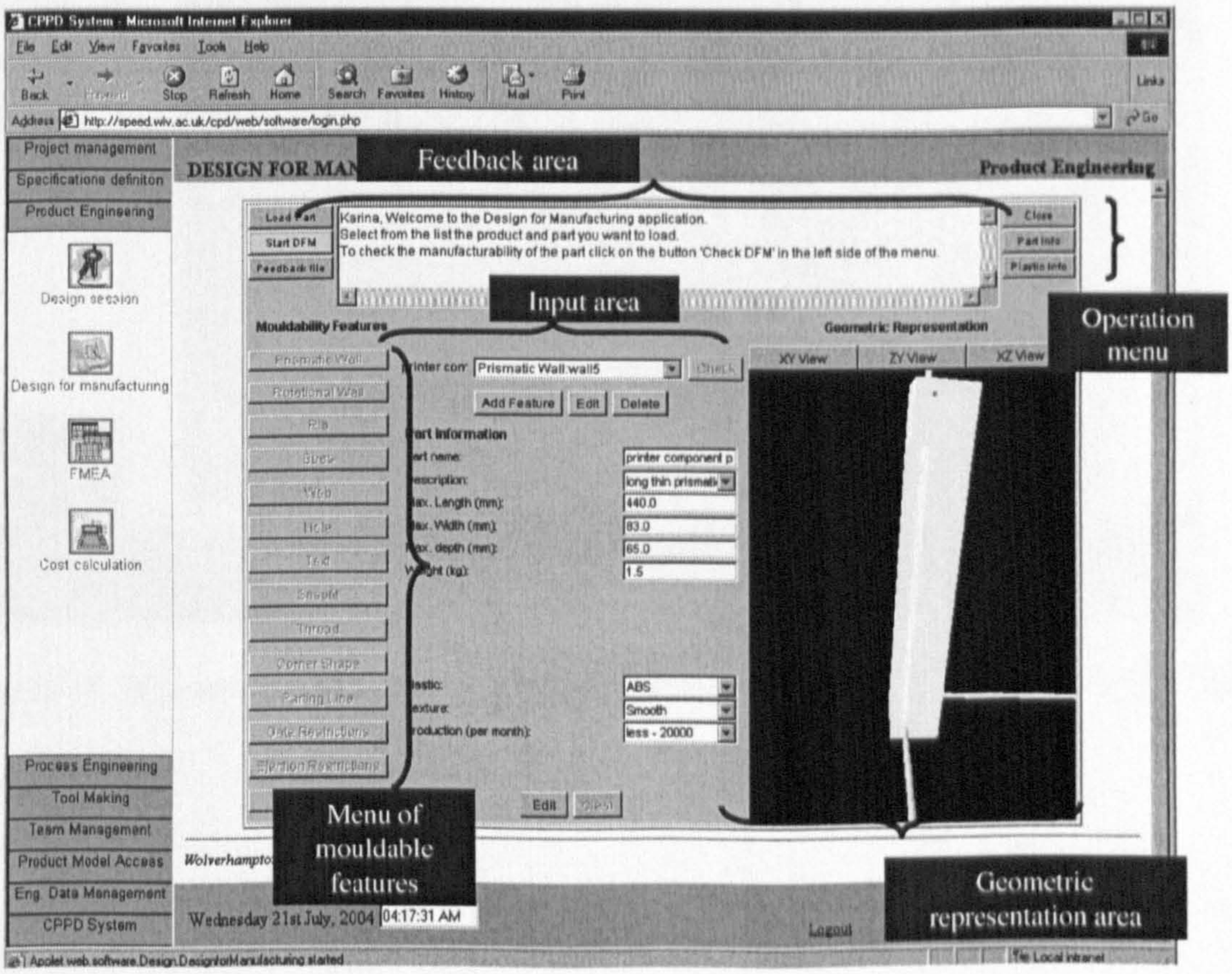


Figure 10.10 Implementation of the graphical user interface of the Design for Manufacturing application

10.2.3 Implementation of "Selection of Production Equipment" class

The graphical user interface of this application is illustrated in figure 10.11. The implementation of this application followed the design model described in section 9.4.2.3. It has the same style as the previous applications containing four main areas: operation menu, feedback area, machines list, and input area.

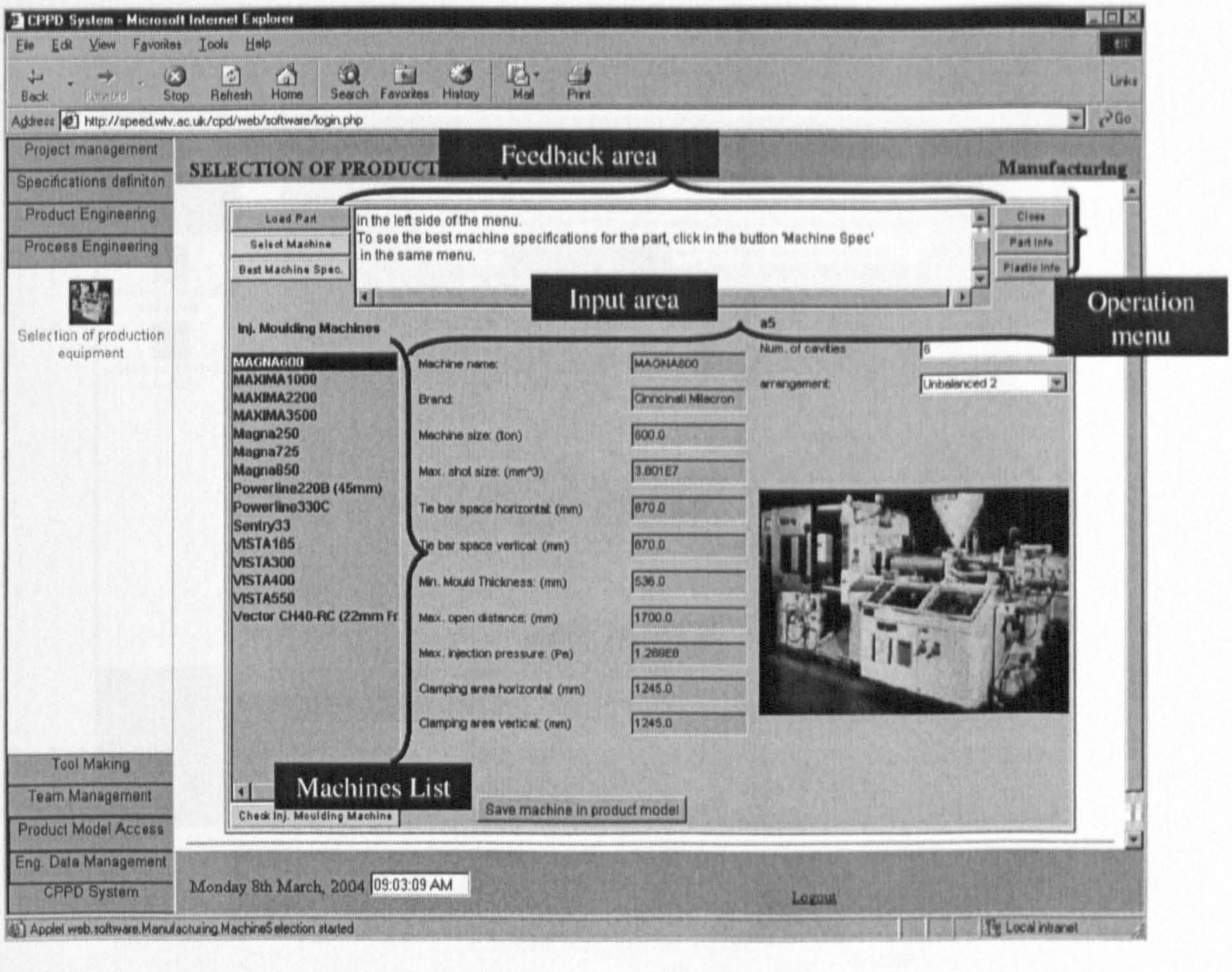


Figure 10.11 Implementation of the graphical user interface of the Selection of Production Equipment application

10.2.4 Implementation of "Selection of Process Parameters" class

This class was implemented as a Java applet following the design models of section 9.4.2.4. Its graphical user interface, shown in figure 10.12, contains three main areas: operation menu, feedback area and input area.

10.2.5 Implementation of "Mould Design and Fabrication" class

This class, which design models were described in section 9.4.2.5, contains a graphical user interface with four main areas: operation menu, feedback area, menu of mould components and input area (see figure 10.13).

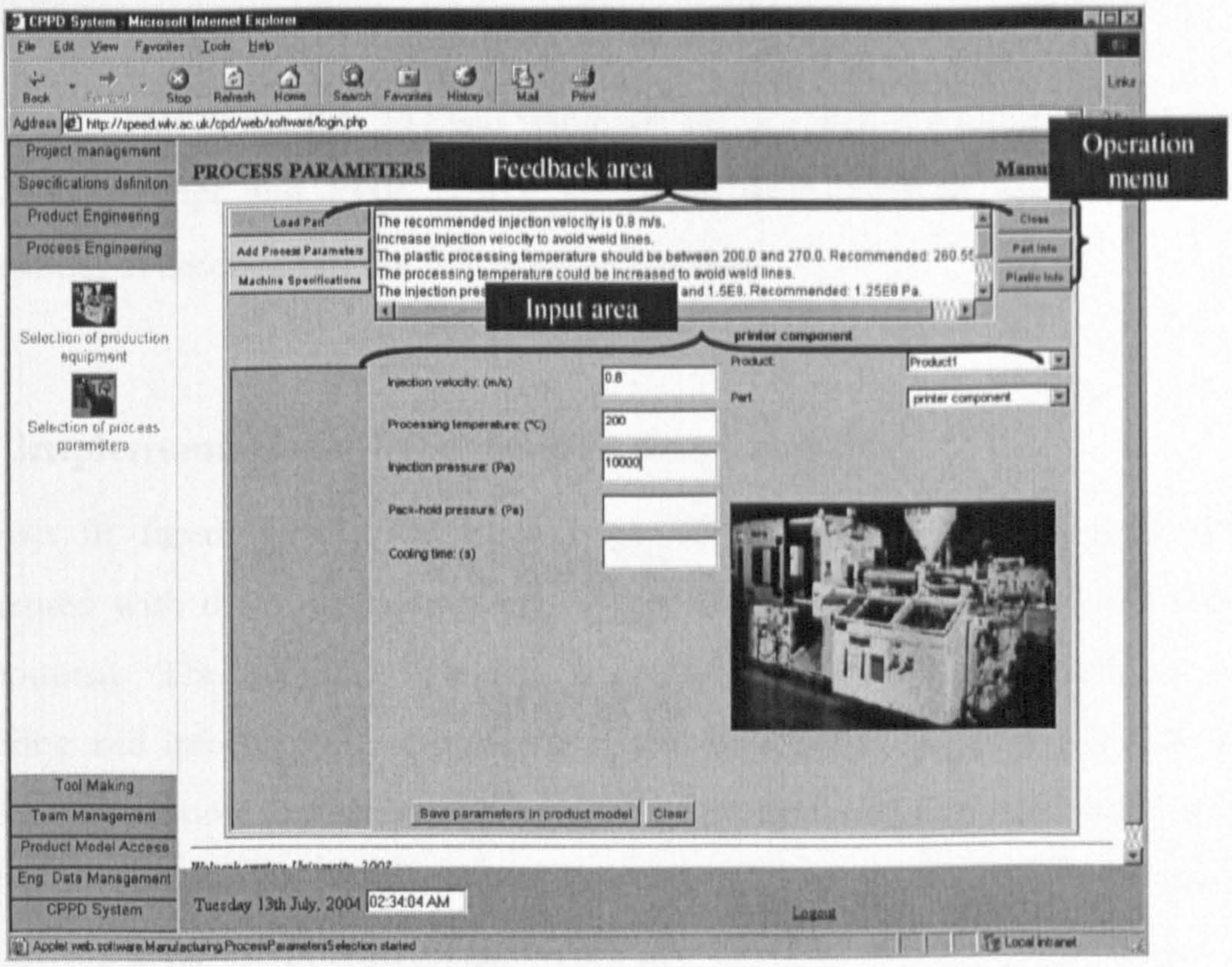


Figure 10.12 Implementation of the graphical user interface of the Selection of Process Parameters application

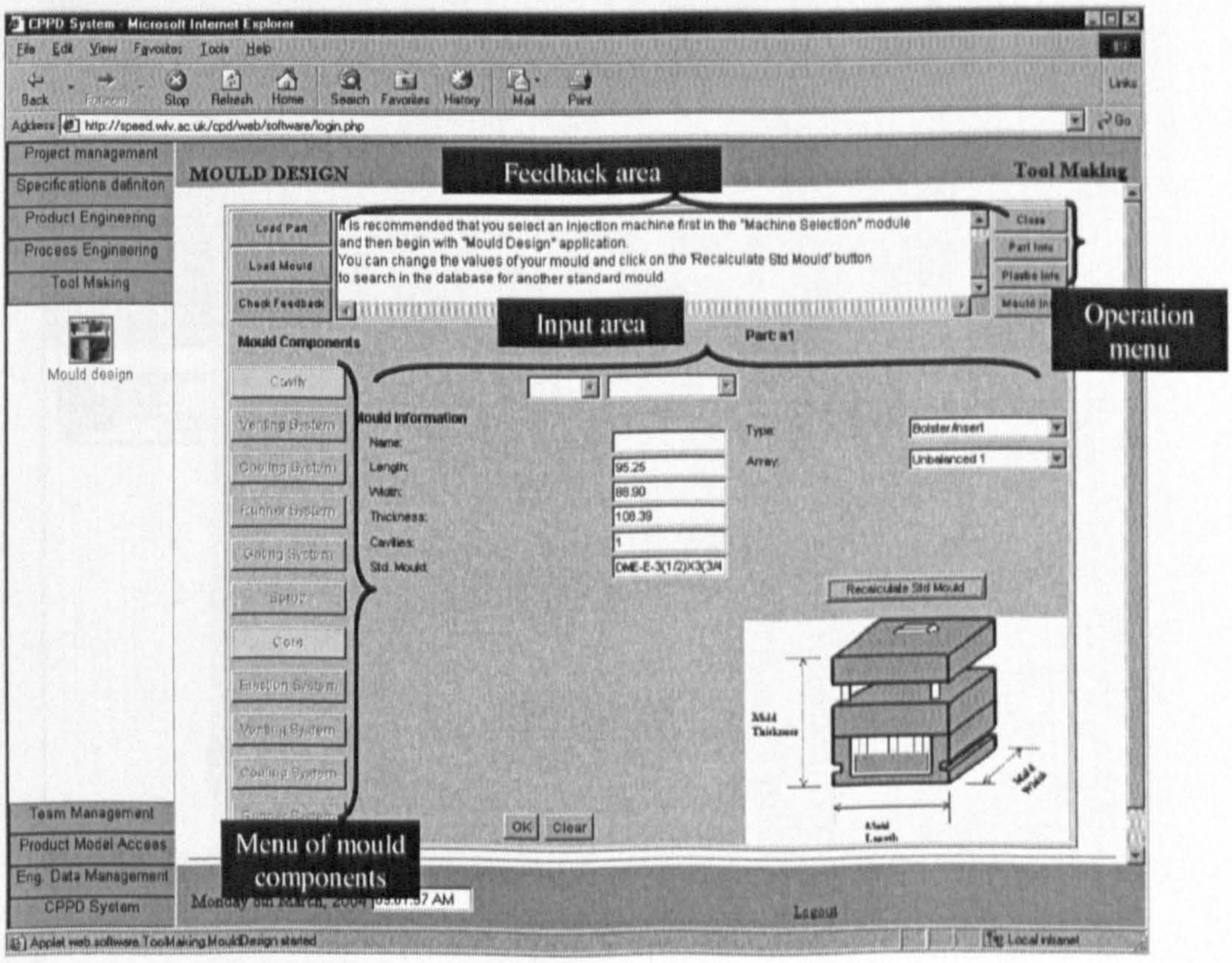


Figure 10.13 Implementation of the graphical user interface of the Mould Design application

10.3 Implementation of the end user layer

The end user layer of the KdCPD was implemented as a web site using languages such as HTML, Javavascript and PHP. These languages were used to implement different functionalities as described in the following subsections.

10.3.1 Implementation of the graphic user interface

As shown in figure 10.14, the main page of the KdCPD system web site was implemented with three main areas: menu (left side), main area (centre) and time/date area (bottom). The menu is divided according to the different decision support engineering and information management applications presented in section 6.4. Within each of these sections a menu of sub applications was implemented using Javascript (Netscape Network 2001-2003) language, which is a scripting language designed for adding interactivity to HTML pages. This allows a submenu of applications to be displayed when one of the menu buttons is selected.

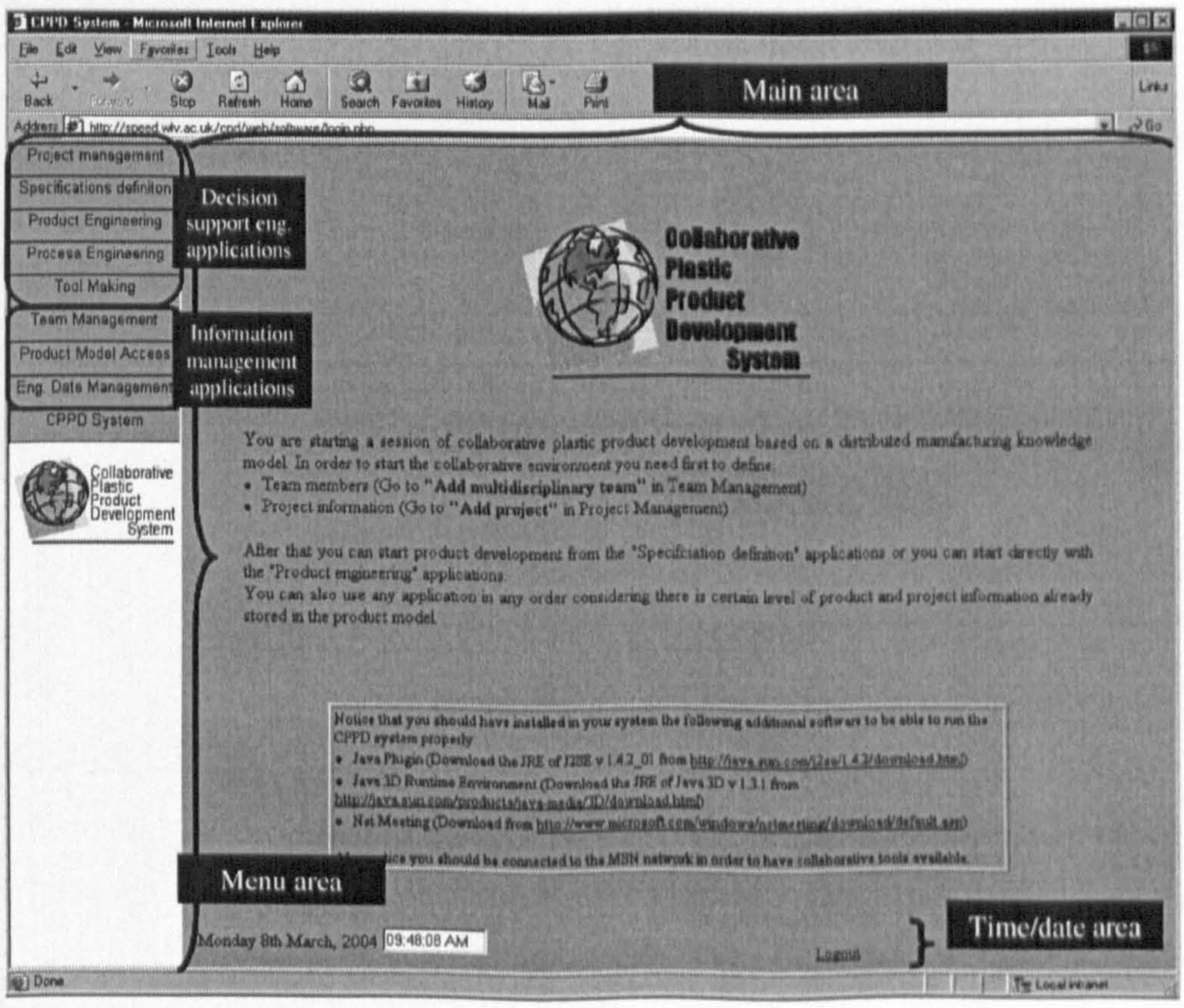


Figure 10.14 Implementation of the graphical user interface of the KdCPD web page

10.3.2 Implementation of the user authentication

In order to provide a secure and customised interface for each of the geographical distributed team members, the user's credentials are requested when logging into the KdCPD system web site. Based on these credentials, the interface is personalised for every different type of user. In order to implement the user authentication, a login web page, illustrated in figure 10.15, was used as an access point for the KdCPD system. A connection was then established with the Product Model database, as this contains the team members' data. The connection was implemented using PHP language and sockets. PHP (*Hypertext Preprocessor*) is an HTML embedded scripting language to write dynamically generated pages (The PHP Group 2001-2004). In addition, a socket is an endpoint for communication between two machines. In the server machine, an application was implemented to listen through one of its ports in order to receive any requests to check the existence of a user. The application functionality was to verify the username and password in the Product Model and to send a response back. In case the user is authorised, the KdCPD system main page is displayed containing only the subapplications for which the team member has access authorisation.

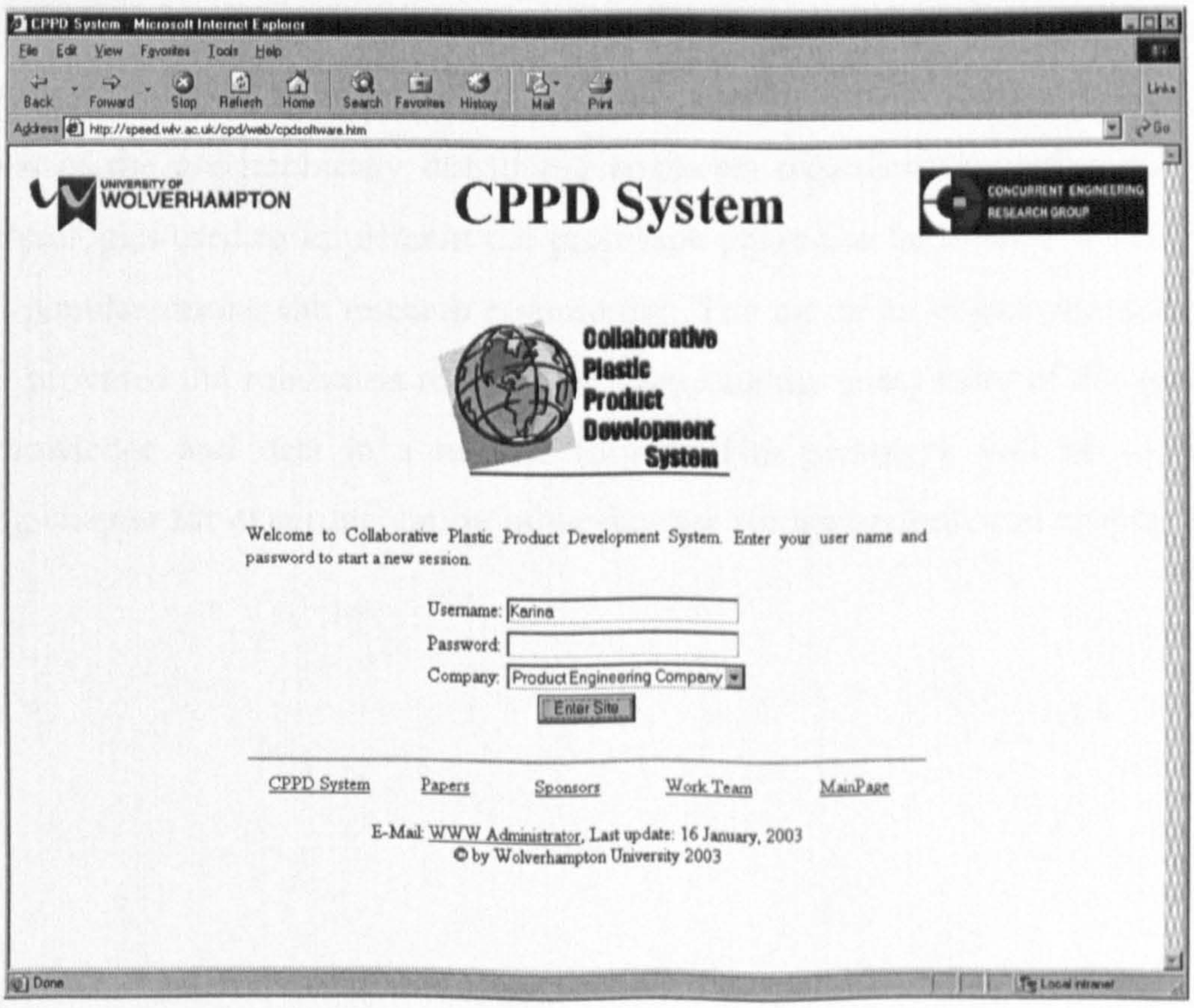


Figure 10.15 Implementation of the login web page of KdCPD system

10.3.3 Implementation of the communication tools

In order to provide a collaborative environment when accessing the engineering applications, an off the shelf technology was selected. NetMeeting (Microsoft Corporation 1996-2000) proved to be a suitable technology and provided advantages, such as free of cost, secure and the availability of real and non-real time communication tools. Such technology can be used to simulate a collaborative environment by starting a NetMeeting session when the geographically distributed team members access any of the decision support engineering applications. NetMeeting provides the following communications tools: chat sessions, videoconference, whiteboard, sharing an application, and email. This software requires to be installed in the users computers and an MSN account need to be obtained from the MSN web site (Microsoft Corporation 1996-2000).

10.4 Closing remarks

This chapter provided a detailed explanation of how the KdCPD system prototype was implemented using object oriented technologies such as Java, Java 3D, Corba and PHP. The system prototype was implemented as a web based system in order to further support the access of the geographically distributed engineers regardless their physical location. The technologies used to implement the prototype proved to be suitable as they are cost free and popular among the research community. The use of an object oriented database manager provided the robustness required by capturing the complexity of the product life cycle knowledge and data in a suitable format. This prototype will be used in the following chapter for experimentation using the case studies presented in chapter 8.

Chapter 11

Knowledge Driven Collaborative Product Development Environment

11.1 Introduction

The experimentation presented in this chapter has been performed to demonstrate how the proposed KdCPD system architecture effectively supports collaborative product development by providing product life cycle knowledge and data in the time, place and format required. The case studies presented in chapter 8 have been used for this experimentation stage.

11.2 Experimentation of the Design for Manufacturing application

During this application, the geographically distributed team members collaborate to consider the manufacturability of the part design. For the experimentation of this application, three case studies (see section 8.2) are presented in the following subsections.

11.2.1 Experiment 1: collaborative design for manufacturing analysis of a prismatic part

This section presents the experimentation done in the KdCPD system prototype using case study 1, which was presented in section 8.2.1. The experiment demonstrates how the collaboration between the product engineer and the process engineer is supported in order to ensure the manufacturability of the batteries' cover prismatic part. Prior to this, the following subsection describes how the system prototype is accessed and how the information of the geographically distributed team is registered in such a collaborative environment.

11.2.1.1 Registration of the geographically distributed team in the KdCPD system

In order to gain access to the engineering applications, the login web page needs to be accessed[1]. This page requests a user name and password in order to enable the provision of a customised graphical user interface. Once the main page is loaded into the browser, a menu of decision support engineering applications is displayed in the left side. The icons in this menu must be clicked in order to gain access to the applications.

Before starting the development of a plastic part in the KdCPD system, the project leader needs to define in the Product Model the data of the geographically distributed team members. This data, which includes name, department, location, telephone and email, is defined in the "Multidisciplinary Team Member Data Management" application (see figure 11.1).

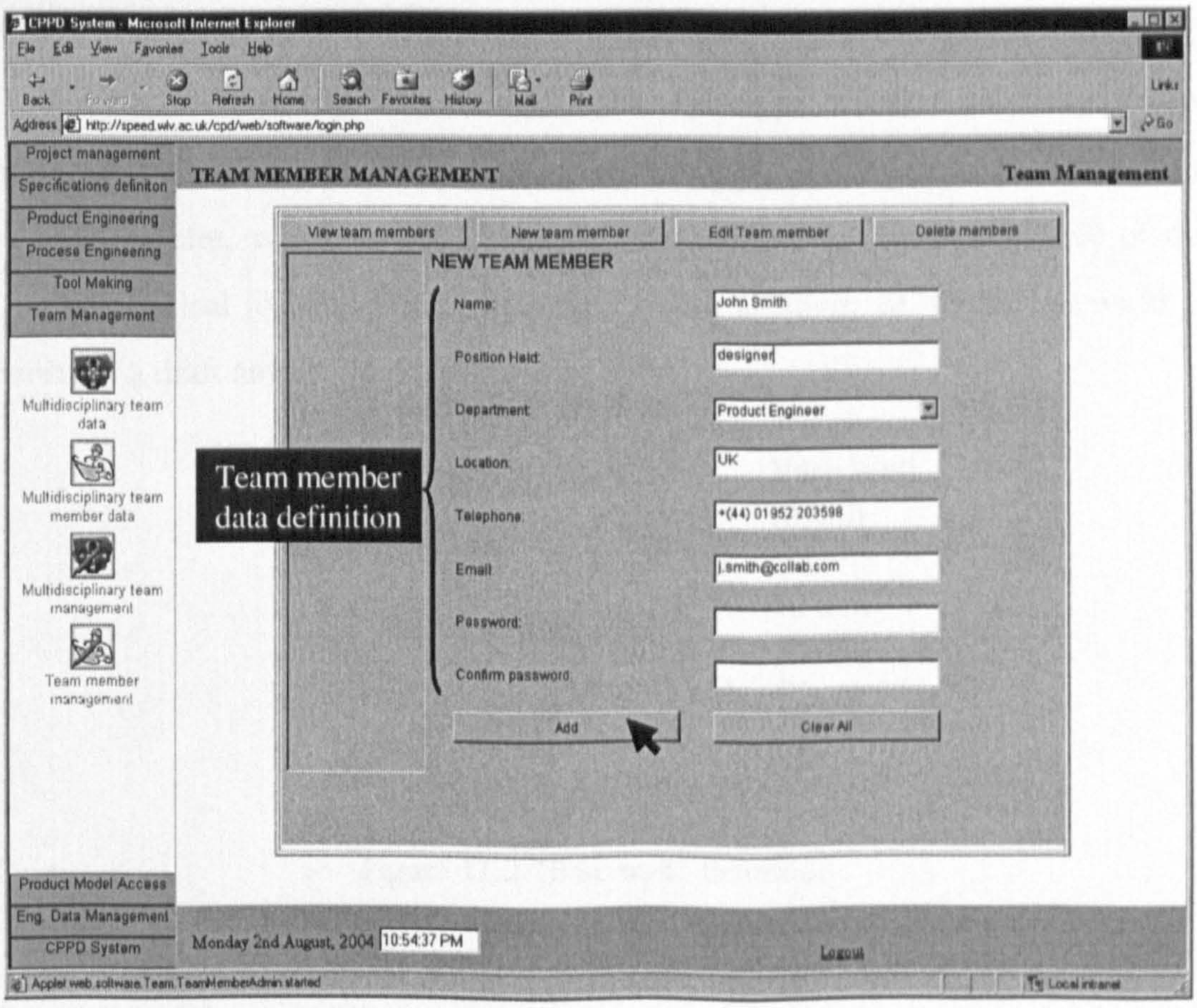


Figure 11.1 Team member data definition in the KdCPD system

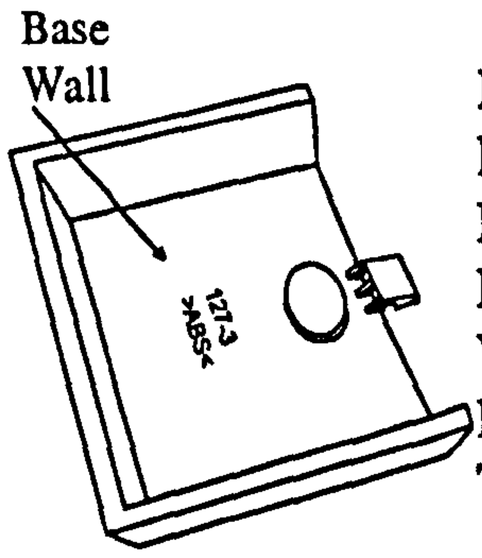
[1] See appendix F for the user guide of the KdCPD system prototype

Thereafter, the project leader sets up a multidisciplinary team and stores its member's names in the Product Model using the "Multidisciplinary Team Data Management" application. Moreover, the data of the project, such as name of the project, customer, and the name of the product is stored along with the name of the multidisciplinary team assigned to the project.

The defined project and multidisciplinary team data is used throughout product development in the KdCPD system. For example, according to the job description, different access rights for the decision support engineering applications are provided for each of the team members.

11.2.1.2 Collaborative design for manufacturing analysis in the KdCPD system

The plastic part used for this experiment is the batteries' cover plastic part. The experiment is started by the product engineer clicking on the Design Session application icon in the KdCPD system. The Design Session then provides an interface, where the product engineer defines the conceptual design in terms of features. An example is the "Base Wall" feature, which is defined in figure 11.2, in the position (0,0,0) of the 3D space, as not critical for the part functionality, with a length of 40 mm, a width of 50 mm, without a draft angle and a thickness of 6 mm.



Name: Base Wall
Position: (0,0,0)
Is Critical: No
Length: 40 mm
Width: 50 mm
Draft angle : 0°
Thickness: 6 mm

Figure 11.2 "Base wall" definition

The feature's definition is filled in the input area of the application, as shown in figure 11.3. The "OK" button is then pressed to store the definition in the Product Model. Afterwards, the 3D virtual geometric model of the feature is displayed in the geometric representation area, as shown in figure 11.3.

The product engineer can review, at any time, a summary of the part's definition by pressing the "Part Info" button. This generates a pop up window with the definition of each of the features, as it has been stored in the Product Model (see figure 11.4).

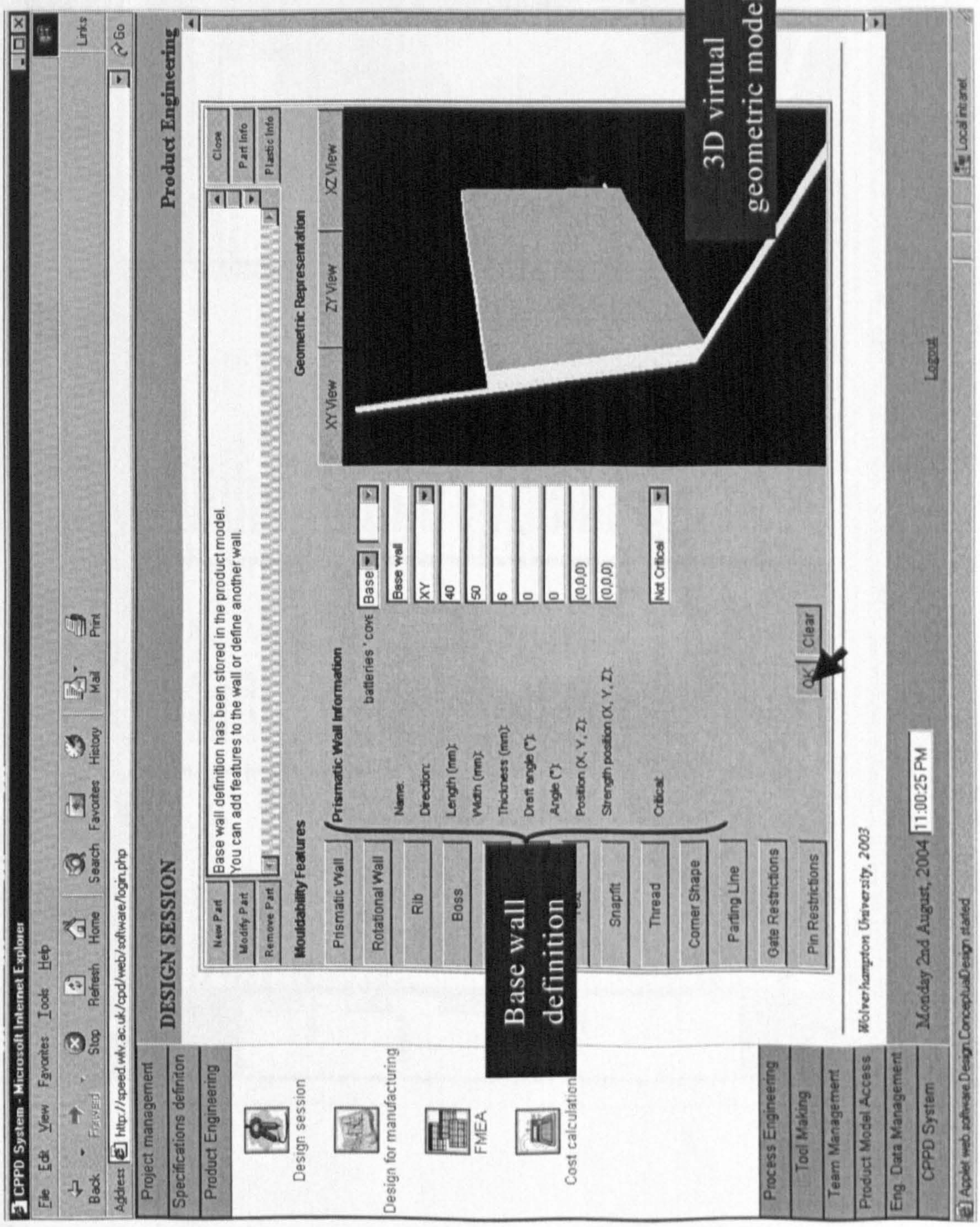


Figure 11.3 "Batteries' cover" plastic part design definition stored through the Design Session application

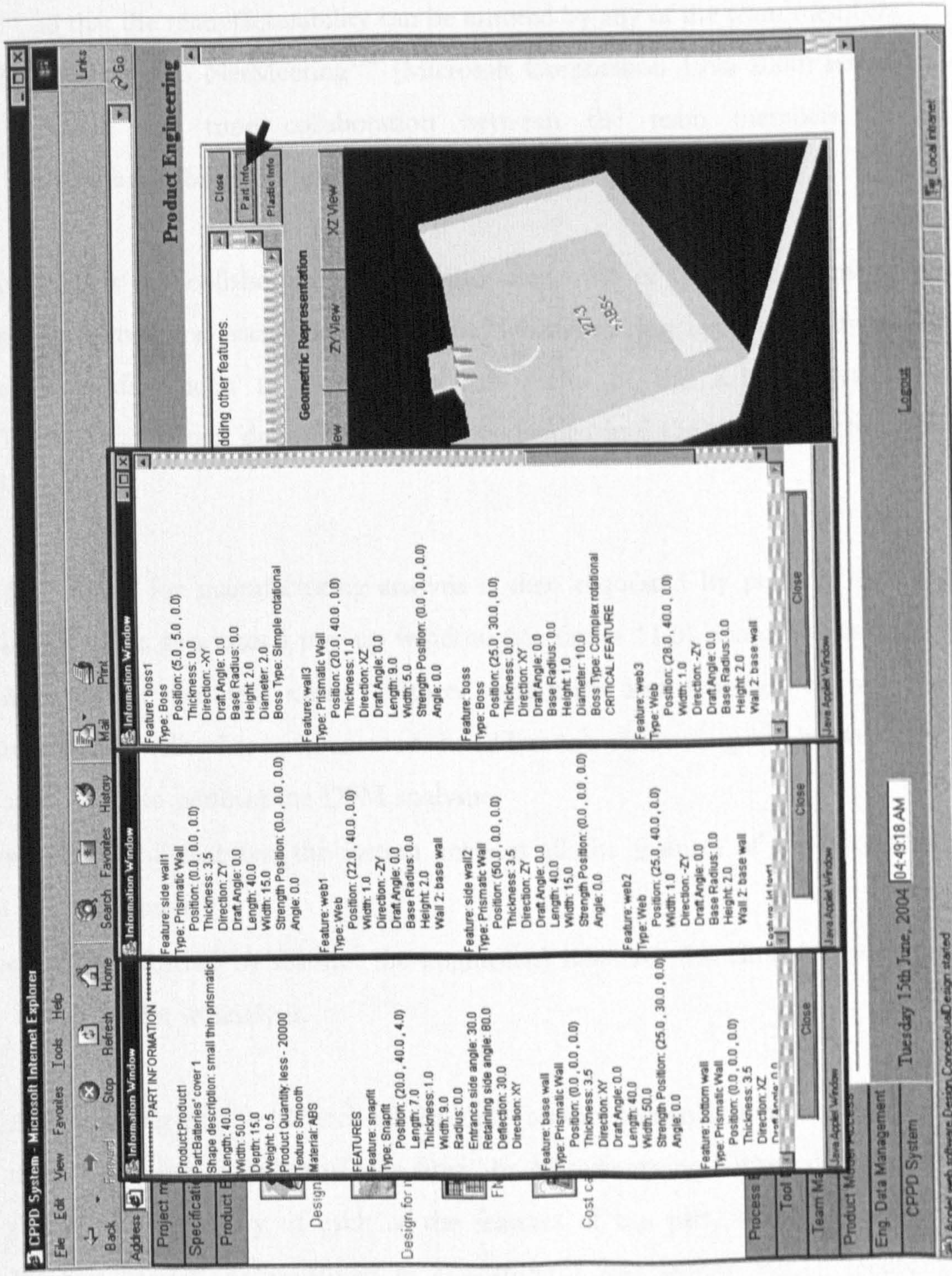


Figure 11.4 "Batteries' cover" plastic part data retrieved from the Product Model through the Design Session application

Thereafter, the product engineer collaborates with the process engineer to ensure the manufacturability of the part. The collaboration is done as follows:

1. By providing feedback advice based on shared product life cycle knowledge and data so that the manufacturability can be ensured by any of the team members.
2. By starting a NetMeeting™ (Microsoft Corporation 1996-2000) session in order to enable real time collaboration between the team members to ensure the manufacturability of the part.

Regardless the collaboration mechanism used, one of the team members, usually the product engineer, starts the Design for Manufacturing application by clicking on its corresponding icon in the application's menu of the KdCPD system web page. Thereafter, the part definition needs to be loaded into the application from the Product Model.

The design for manufacturing analysis is then requested by pressing the "Start DFM" button. This generates a pop up window (see figure 11.5), which contains the following information: features' name, features' criticality and whether or not the feature's manufacturability has been ensured. In addition, a choice is given between two different approaches to perform the DFM analysis:

- Check all features: the system analyses all the features of the part prioritising the critical ones.
- Check feature by feature: the engineer(s) has/have the choice to select any specific feature for its analysis.

After clicking on the "Check all features" button, the analysis is performed by invoking the product life cycle knowledge from the Manufacturing Knowledge Model to validate the manufacturability of each of the features of the part, which data is stored in the Product Model. As explained in case study 1 (see section 8.2.1), feedback advice is generated and it is displayed by the application in a feedback pop up window. This window can be scrolled down and up to review all the feedback. The feedback is also stored in a file to be shared amongst the geographical distributed team. Figure 11.6 illustrates three screenshots of the pop up window containing the feedback. Only the top window has been enlarged to make clearer the presentation of the text.

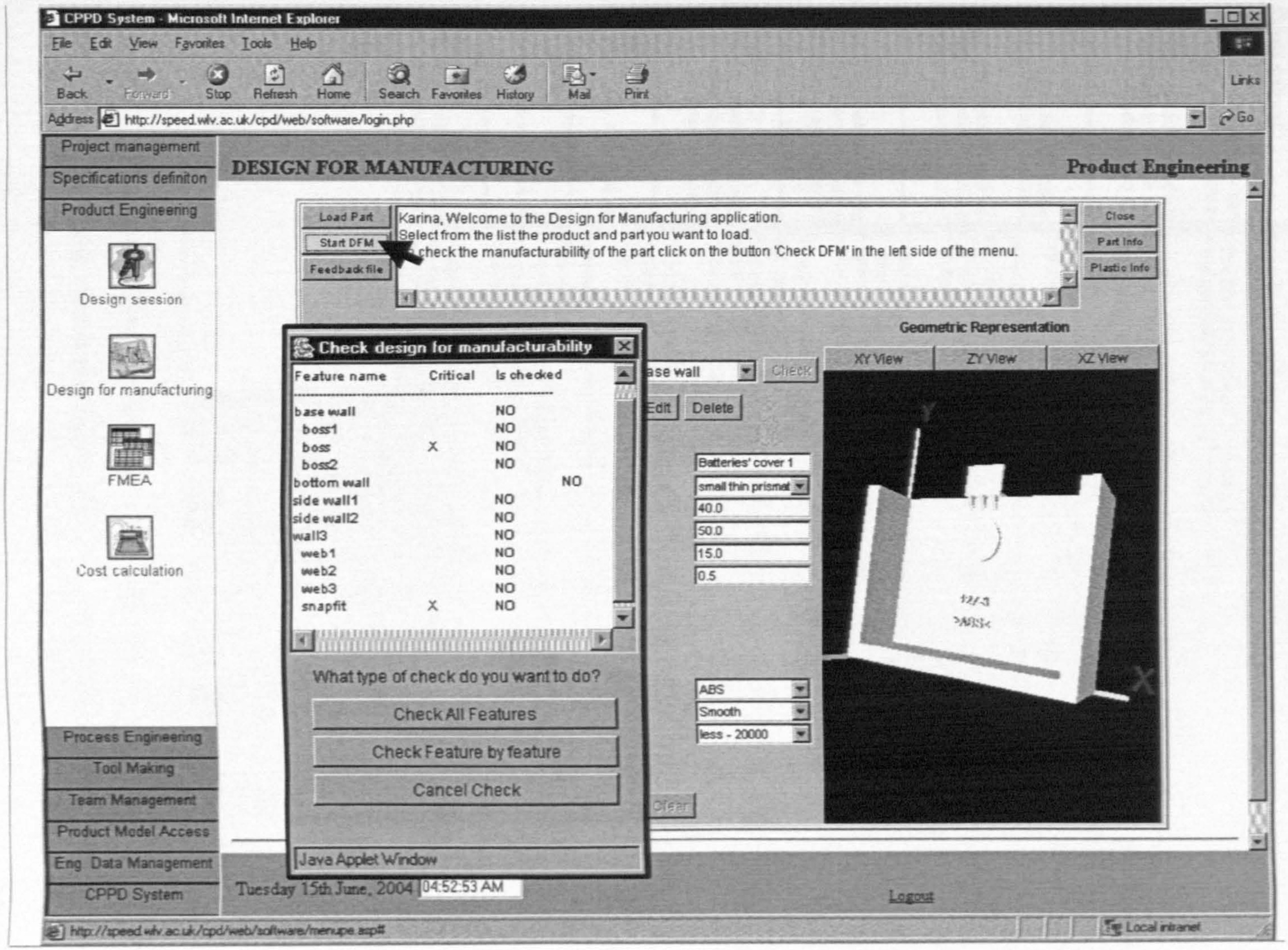


Figure 11.5 Invoking the DFM analysis of the "Batteries' cover" plastic part in the Design for Manufacturing application

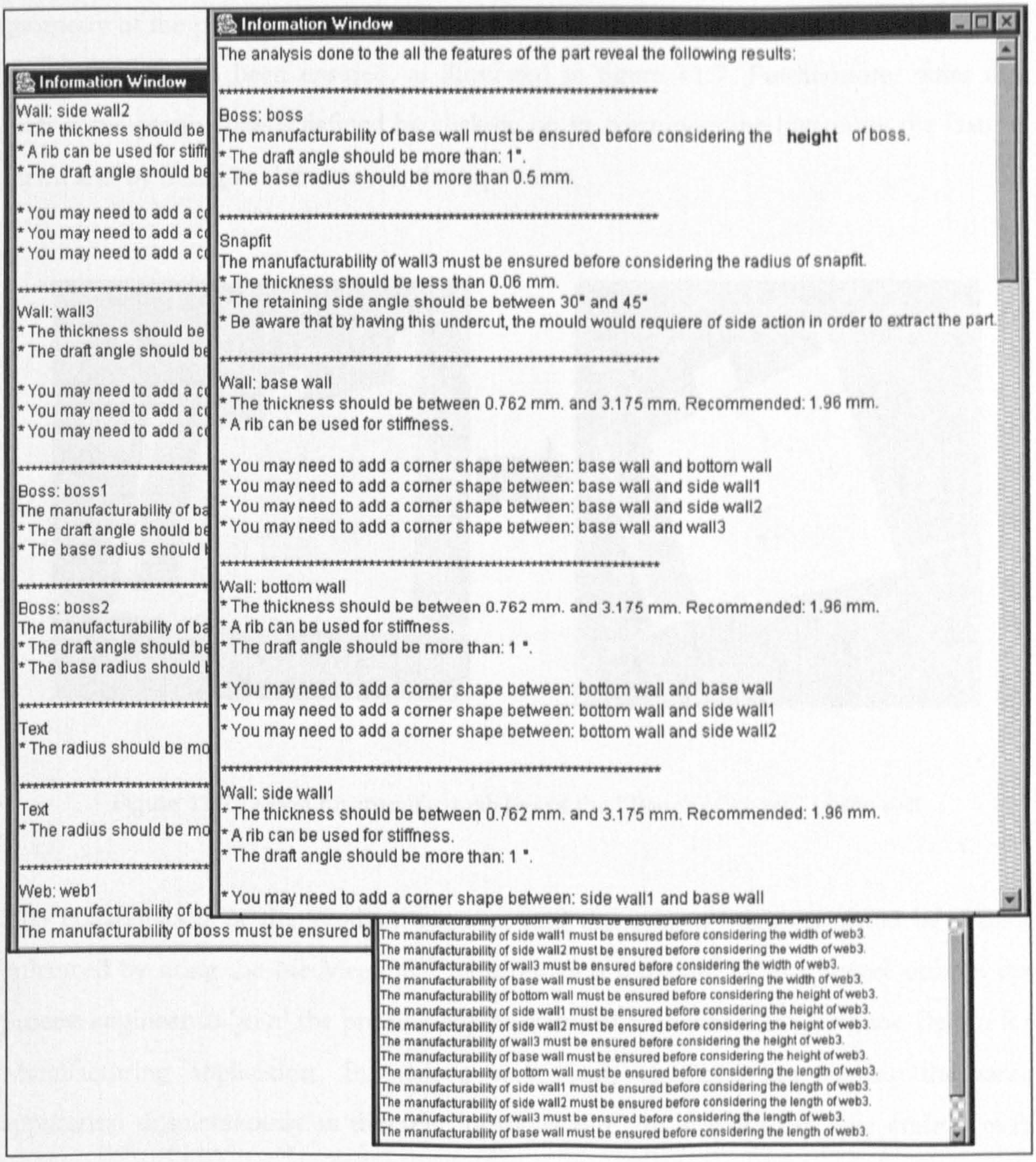


Figure 11.6 Feedback advice provided by the Design for Manufacturing application regarding the manufacturability of the "Batteries' cover" plastic part

After receiving the feedback advice, the product engineer needs to change the definition of the features in the appropriate fields of the input area according to the advice. The new definition is then stored in the Product Model by pressing the "OK" button. At the same time, the system displays these changes in the virtual geometric model. In such way, the team members are aware of how the manufacturing constraints directly affect the

geometry of the part. The DFM analysis can be called again until the manufacturability of all the features has been ensured, as illustrated in figure 11.7. Furthermore, other data such as the parting line is defined by clicking on its corresponding button on the features menu and by filling its definition in the input area.

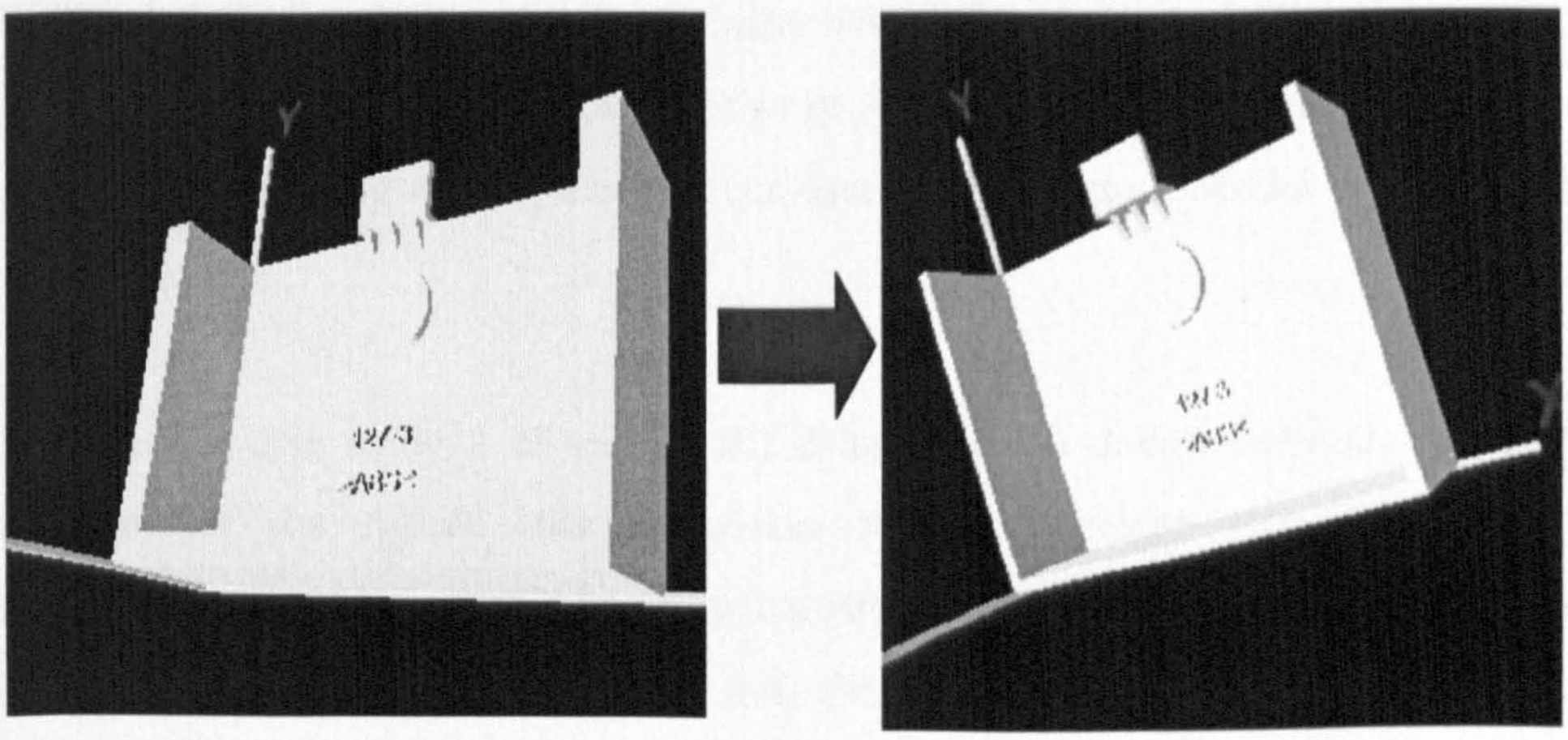


Figure 11.7 Design for manufacturability of the "Batteries' cover" plastic part

The collaboration of the product engineer and the process engineer could be further enhanced by using the NetMeeting™ collaboration software[2]. This software enables the process engineer to 'join' the product engineer in real time when accessing the Design for Manufacturing application. In such a way, both engineers are sharing the same application simultaneously in different locations and any of them can take control over the application at any time. Other tools, such as chat or videoconference are available for discussion regarding the modifications to the part definition.

This experiment has shown how the KdCPD system supports the collaboration between the geographically distributed team members by providing product life cycle knowledge and data to support the design for manufacturability of a plastic part.

---

[2] See appendix F for the user guide of the NetMeeting software

11.2.2 Experiment 2: invoking the DFM analysis as a request of other engineering application

This section presents the experimentation done in the KdCPD system prototype using case study 2, which was presented in section 8.2.2. In the experiment, the toolmaker collaborates with the product engineer when designing an injection mould for a rotational plastic part which manufacturability has not been ensured. The plastic part used for this experiment is the liquid container's cap plastic part. The experiment is started by the product engineering storing the product data in the Product Model using the Design Session application.

As described in case study 2 of section 8.2.2, in order to design correctly some of the components of the mould, the toolmaker needs to ensure that the part under consideration is within manufacturability constraints. The KdCPD system supports the collaboration between the toolmaker and the product engineer by supporting the toolmaker to determine if the manufacturability of the plastic part has been ensured. This is done through the Mould Design and Fabrication application, which is accessed by clicking on the corresponding icon in the application menu of the KdCPD system web page. Once the application is accessed, the next step is to load the plastic part definition from the Product Model by selecting its name from a list. When the toolmaker clicks on one of the component's button (i.e. cooling system) of the mould components menu, automatically the knowledge integrity of the model highlights the need to check the manufacturability of the part. Therefore, the manufacturability of each of the features is validated as explained in case study 2 (see section 8.2.2). This is done transparently by the KdCPD system despite the fact that the knowledge related to the design features manufacturability constraints is residing in the product engineering site. In this case study, the manufacturability of the plastic part has not been ensured. Therefore, the message shown in figure 11.8 is displayed.

After receiving the feedback, the toolmaker collaborates with the product engineer in either of the following ways:

1. By the toolmaker accessing the Design for Manufacturing application to request feedback advice based on shared product life cycle knowledge and data to ensure the part manufacturability. The toolmaker then follows the advice (see figure 11.9) and modifies the part definition in the product Model.

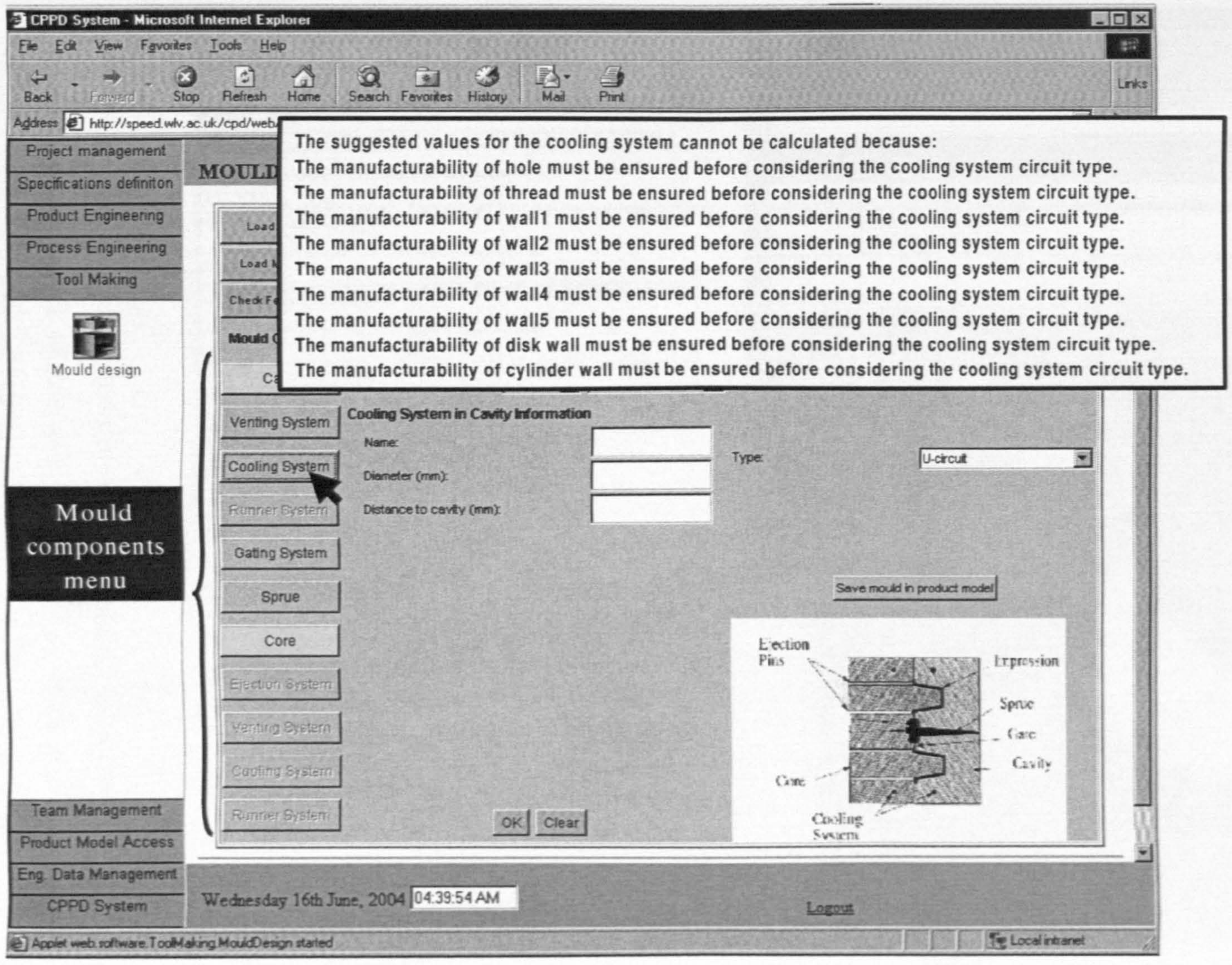


Figure 11.8 Feedback advice provided by the Mould Design and Fabrication application regarding the need to ensure the manufacturability of the part before considering the cooling system in the injection mould

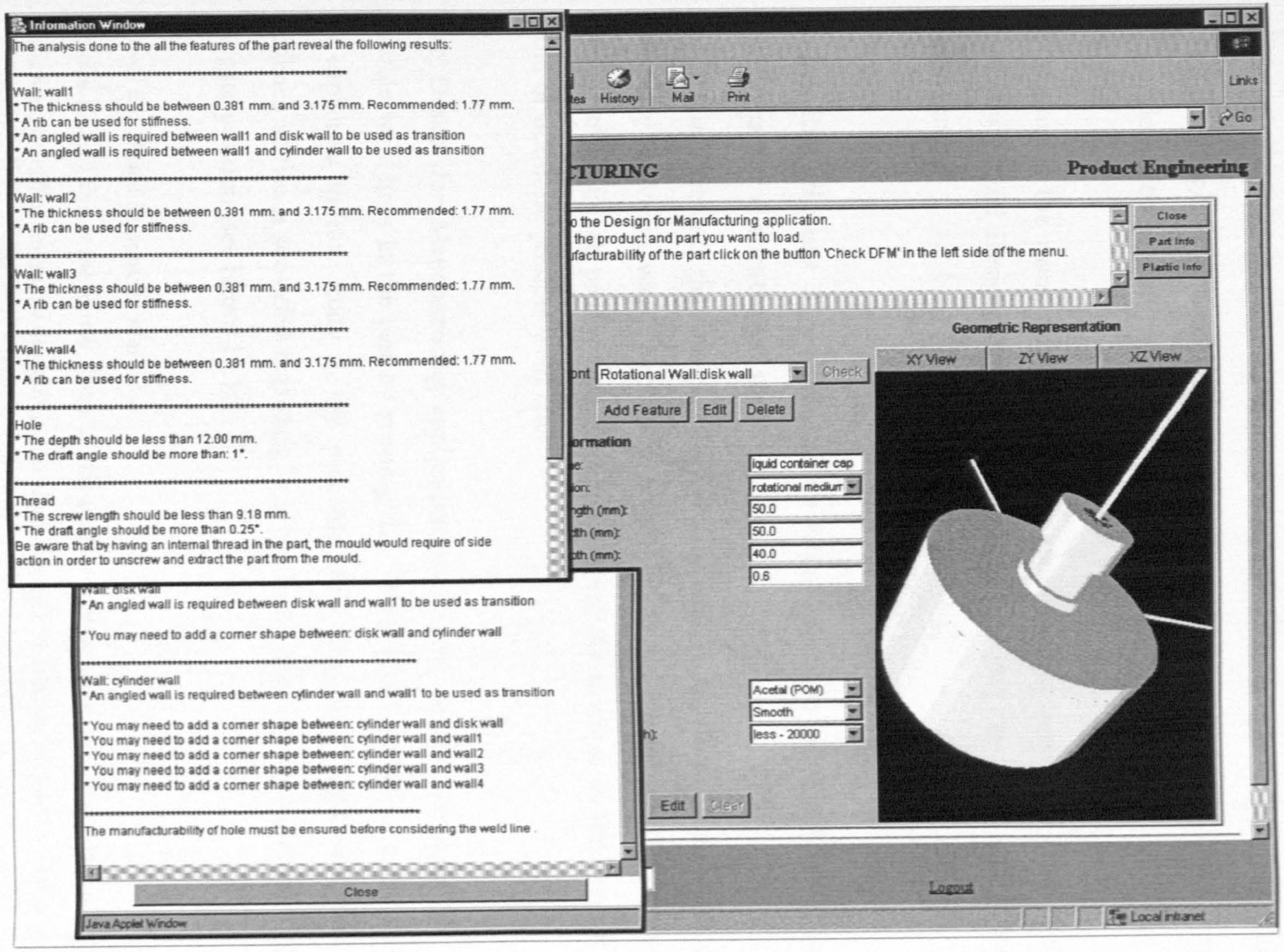


Figure 11.9 Feedback advice provided by the Design for Manufacturing application regarding the manufacturability of the "Liquid container cap" plastic part

2. By both toolmaker and product engineer using NetMeeting™ to simultaneously access the Design for Manufacturing application. Both engineers then collaborate using communication tools to modify the part definition in the Product Model.

3. By the toolmaker using NetMeeting™ communication tool to inform the product engineer of the need to ensure the manufacturability of the part. Thereafter, the product engineer accesses the Design for Manufacturing application and modifies the part definition in the Product Model.

After ensuring the manufacturability of the part, its data is accessed from the Product Model by Mould Design application in order to support the definition of the mould components.

11.2.3 Experiment 3: collaborative identification of weld lines and gate positions for a rotational part

This section presents the experimentation done in the KdCPD system prototype using case study 3, which was presented in section 8.2.3. In the experiment, the toolmaker collaborates with the product engineer to optimise the location of the gates in order to avoid or minimise possible weld lines.

The Design for Manufacturing application guides the product engineer to recognise possible weld lines in the part by pressing the "Weld Line" button in the features menu. As explained in case study 3, the probable weld lines in the part are identified and feedback advice is provided regarding how the weld lines can be reduced by optimising the gate position (see figure 11.10):

A weld line should be expected near:

Position (0,5,0) due to the feature "Boss". This weld line can be reduced by: placing the gate position close to the boss, providing venting at the weld line or increasing melt or injection temperature during production.

Following the advice, the product engineer presses the "Gate Restrictions" button in the features menu to request feedback advice regarding the suitable positions for the gates. As explained in case study 3 (see section 8.2.3), the product life cycle knowledge is transparently invoked by the KdCPD system despite the fact that this specific knowledge is residing in the tool making site. The following feedback is provided (see figure 11.11).

A suitable position for the gate is one of the following:

- A non visible surface area.

- "Cylinder Wall", where the gate is farthest from "Boss"

- The centre of "Disk Wall", where the material flow will have an even distribution.

- Be aware that weld lines are expected when two flows meet.

A restriction area could be placed to specify to the toolmaker an unwanted or preferred gating position.

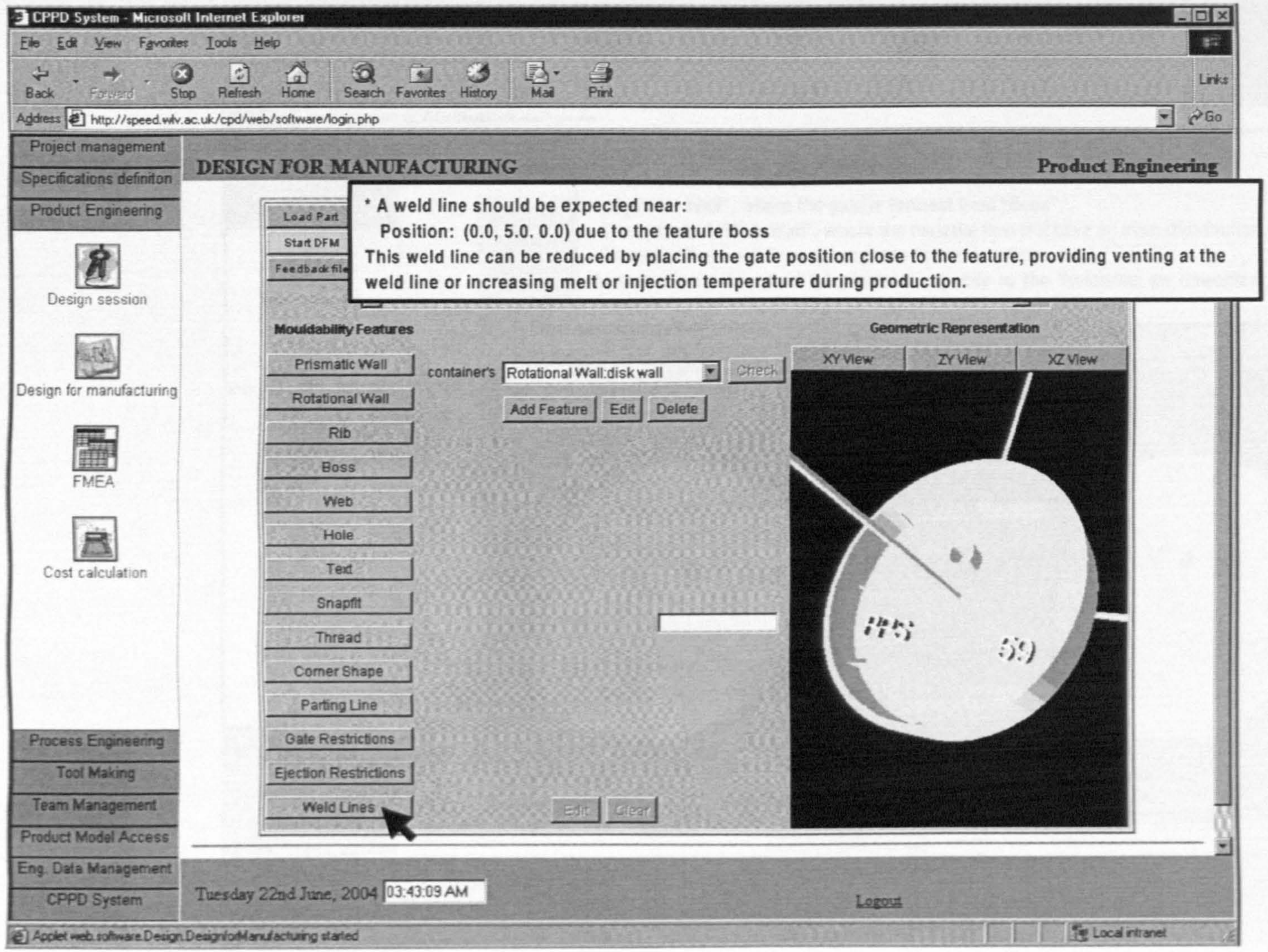


Figure 11.10 Feedback advice provided by the Design for Manufacturing application regarding the possible location of weld lines

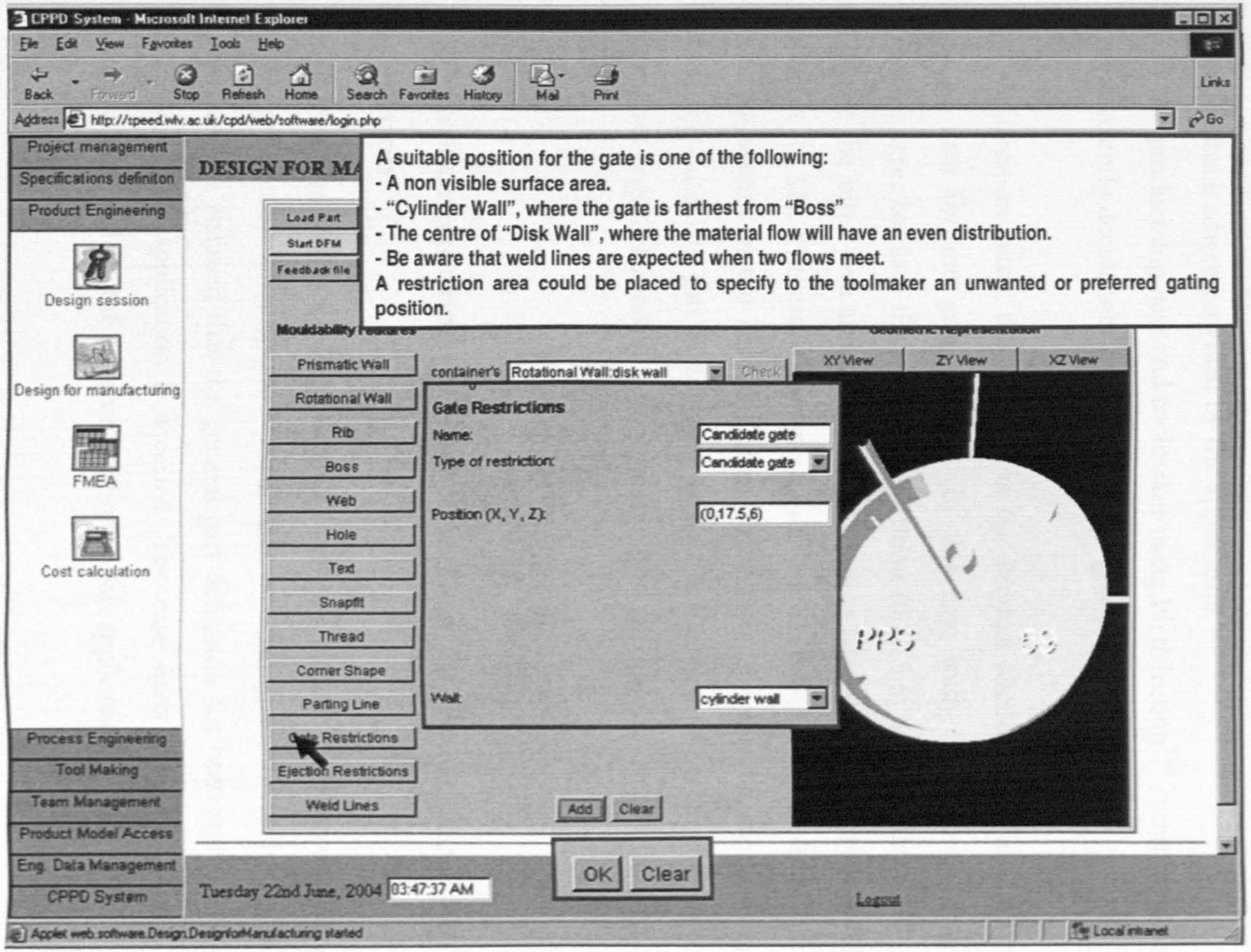


Figure 11.11 Feedback advice provided by the Design for Manufacturing application regarding the suitable locations for gates in the "Container cap" plastic part

Based on the feedback the product engineer collaborates with the toolmaker to decide on the gate positions. This is achieved as follows:

1. By the product engineer deciding whether or not a gating mark is preferred based on the feedback advice provided by the application.
2. By the product engineer and toolmaker using NetMeeting™ communication tools to concurrently decide which are the suitable gate positions.

After the decision taken, this is stored in the Product Model by defining a candidate or restricted area for the gate position. In this case study, the product engineer and toolmaker agree to place the gate in the position (0,17.5,6) of "Cylinder Wall". This data is filled in the input area and stored in the Product Model by pressing the "OK" button (see figure 11.11). The candidate gate is later retrieved when the toolmaker is designing the gating system in the Mould Design application. By pressing the "Gating System" button, the candidate gate is displayed as feedback advice to ensure that the opinion of the product engineer is taken into account when placing the gate in the part (see figure 11.12).

11.3 Experimentation of the Selection of Production Equipment application

The process engineer uses this application to select a suitable injection moulding machine for the production of a specific plastic part according to the process and resource limitations. It is assumed that the general part definition has been stored in the Product Model when this application is accessed. The case study presented in the following subsection was used for the experimentation of this application.

11.3.1 Experiment 4: collaborative selection of production equipment

This section presents the experimentation done in the KdCPD system prototype using case study 4, which was presented in section 8.3.1. The part used for this experiment is the connector's cap plastic part. This experiment is demonstrating how the collaboration

between the process engineer, toolmaker and product engineer is supported during the selection of a suitable machine for production.

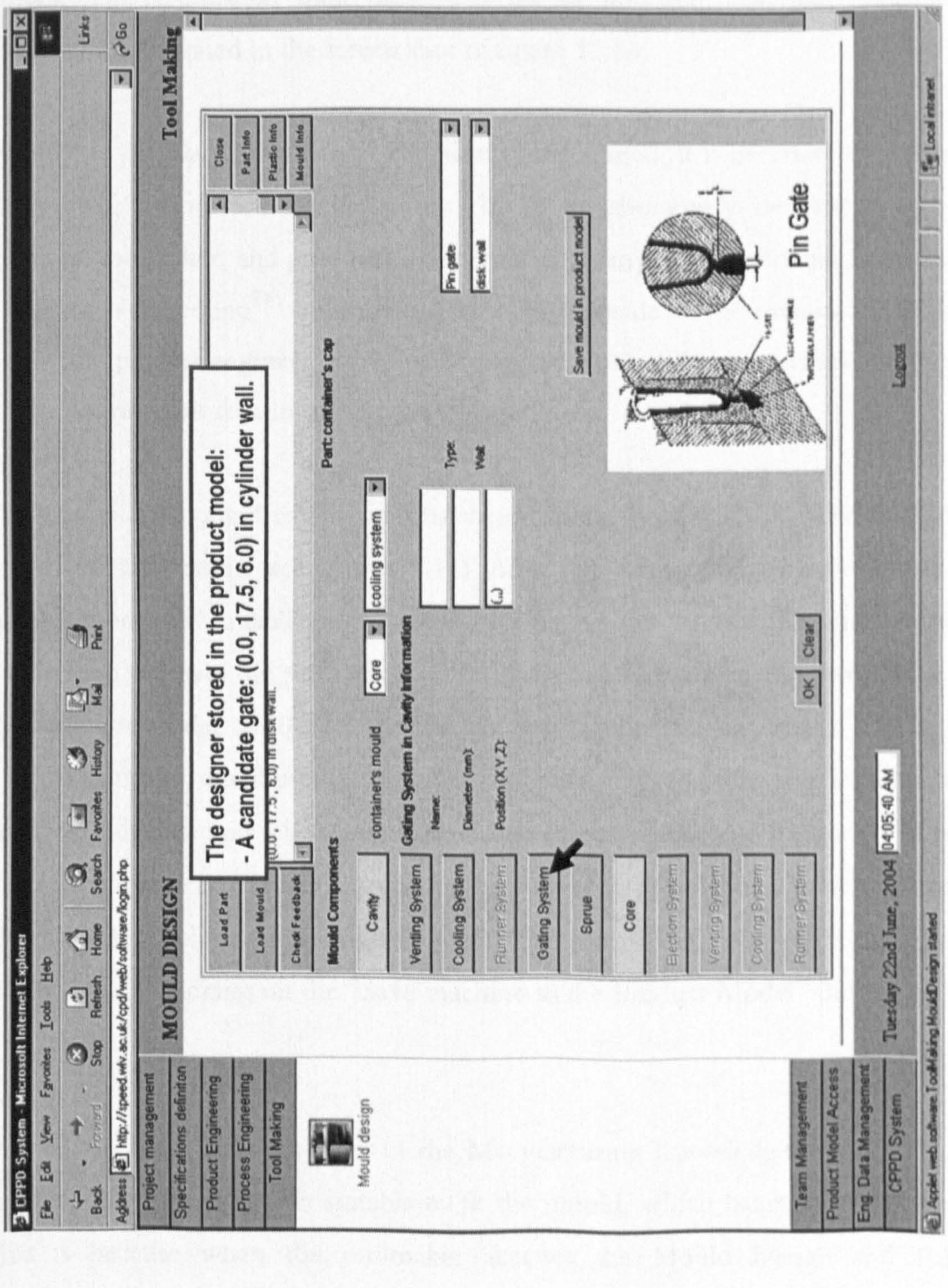


Figure 11.12 Feedback advice provided by the Mould Design and Fabrication application regarding the candidate position for the gate in the "Container cap" plastic part

An injection machine can be selected once a minimum of part data has been defined in the Design Session application, as illustrated in figure 11.13. The Selection of Production

Equipment application is accessed by clicking on its corresponding icon in the KdCPD system web page. Once accessed, the first step is to load the part definition into the application by selecting the part's name from a list of parts available in the Product Model. The application then presents a list of injection machines available in the company as illustrated in the screen shot of figure 11.14.

In order to calculate the machine characteristics required, it is necessary to determine the number of cavities and their layout. This is collaboratively decided by the process engineer, toolmaker, and product engineer by using any of the following mechanisms:

- Using NetMeeting™ communication tools to decide on the cavities and their layout.
- By the process engineer individually deciding the number of cavities and their layout and storing this data in the Product Model.

The choice of having 4 cavities in a balanced arrangement type 1 is selected in the input area of the application (see figure 11.14). After doing this, the process engineer requests the selection of a suitable machine by clicking on the "Select machine" button. The system then retrieves the part definition from the Product Model and invokes the product life cycle knowledge for the selection of the best production equipment. This knowledge is geographically distributed among the different collaborators and it is transparently shared to calculate and select the most suitable injection machine from a list of resources. The selected injection machine, which in this case is VISTA 165, is presented as feedback advice by the application (see figure 11.14). The process engineer selects this machine for production by clicking on the "Save machine in the Product Model" button in the input area.

Due to the knowledge integrity of the Manufacturing Knowledge Model, it is ensured that the selected machine is suitable to fit the mould, which has not been designed yet. This is because when the toolmaker accesses the Mould Design and Fabrication application, the dimensions of the machine are automatically retrieved from the Product Model, and mould plates dimensions suitable for the injection machine are suggested to the toolmaker. By sharing the data and knowledge of the companies, the collaboration

between the toolmaker, process engineer and product engineer is achieved and therefore further iterations are eliminated.

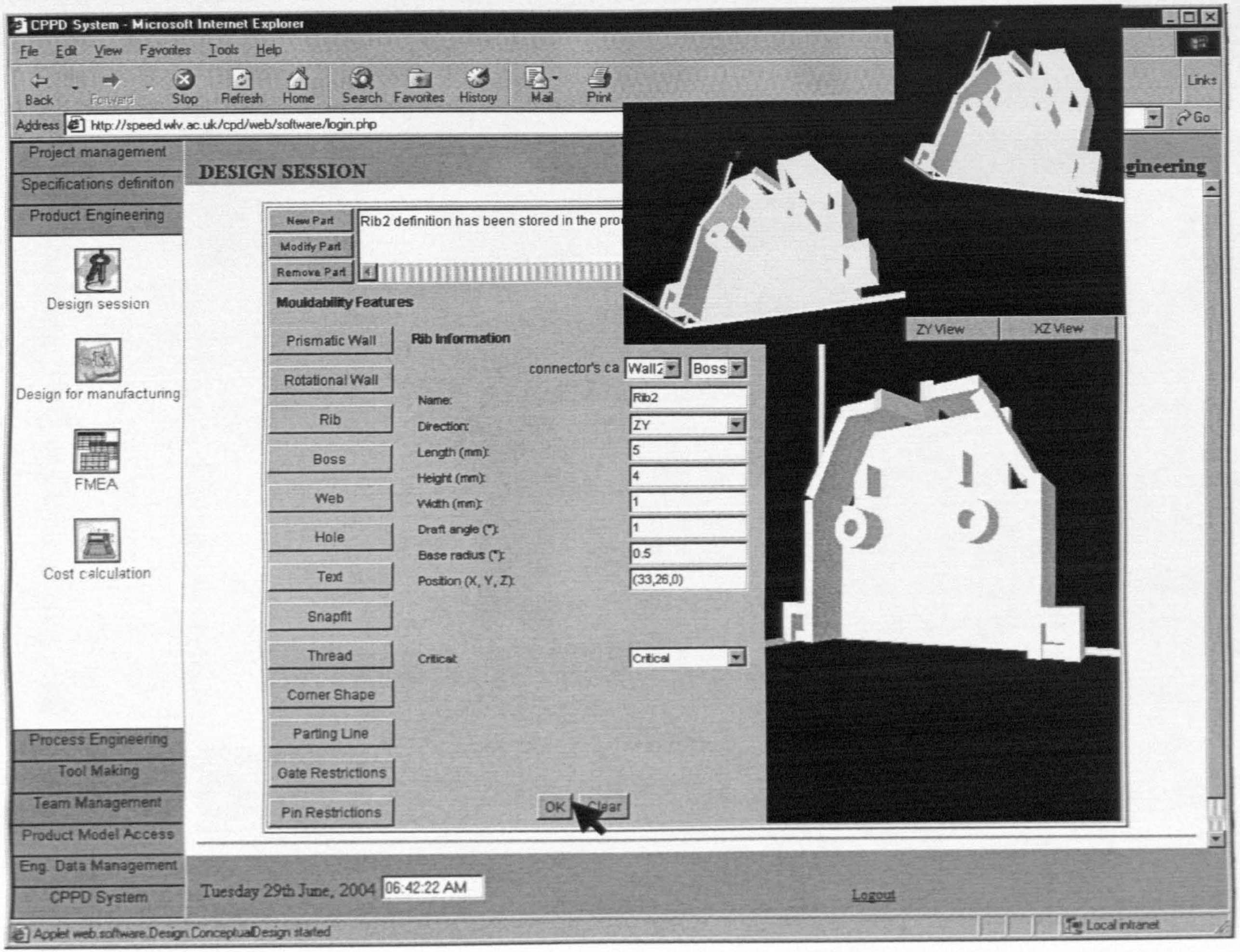


Figure 11.13 "Connector's cap" plastic part design definition stored through the Design Session application

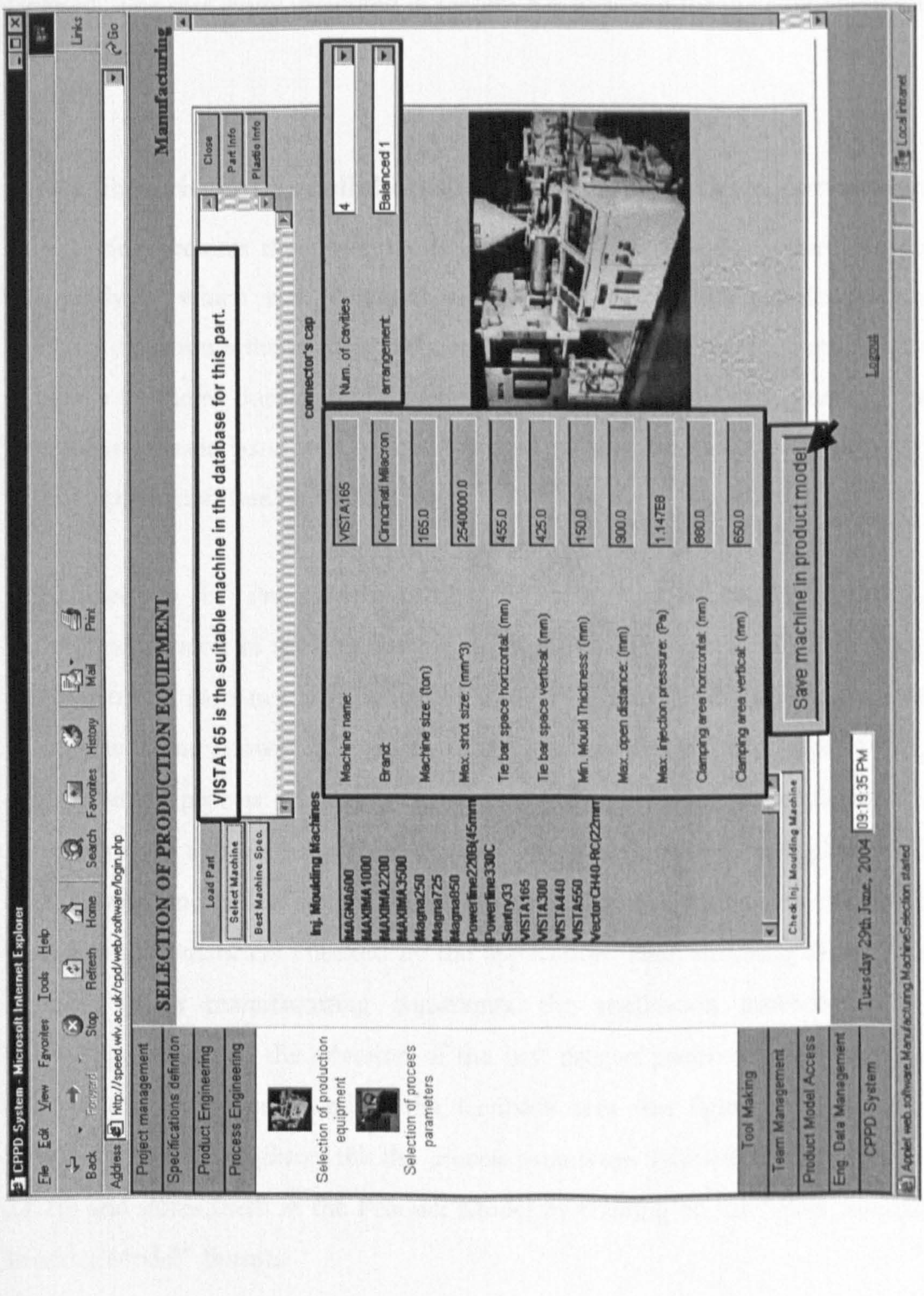


Figure 11.14 Feedback advice provided by the Selection of Production Equipment application regarding the suitable machine to produce the "Connector's cap" plastic part

11.4 Experimentation of the Selection of Process Parameters application

The process engineer uses this application to select the suitable process parameters according to the process and resources limitations. It is assumed that the general definition of the part has been stored in the Product Model when this application is

accessed. The case study presented in section 8.4 was used for the experimentation of this application.

11.4.1 Experiment 5: collaborative selection of process parameters

This section presents the experimentation done in the KdCPD system prototype using case study 5, which was presented in section 8.4.1. In this experiment, the process engineer determines the process parameters for the injection machine in order to avoid or minimise problems during production. The part used for this experiment is the printer's component plastic part, which data is stored in the Product Model using the Design Session application (see figure 11.15).

The Selection of Process Parameters application is accessed by clicking on its corresponding icon in the engineering applications menu of the KdCPD system. Firstly, the data of the parts needs to be loaded into the application by selecting it from a list of parts. Due to the knowledge integrity of the Manufacturing Knowledge Model, decisions such as which process parameters to use cannot be done unless the manufacturability of the part's walls and holes has been ensured. By sharing and providing knowledge related to the walls and holes design constraints in real time, the manufacturability of these features is automatically checked by the application. After ensuring that the features are defined within manufacturing constraints, the application proceeds to invoke the knowledge related to the selection of the best process parameters. The application then displays the advice produced in the feedback area (see figure 11.16). Following this advice, the process engineer fills the process parameters values in the input area (see figure 11.16) and stores them in the Product Model by clicking on the "Save parameters in the Product Model" button.

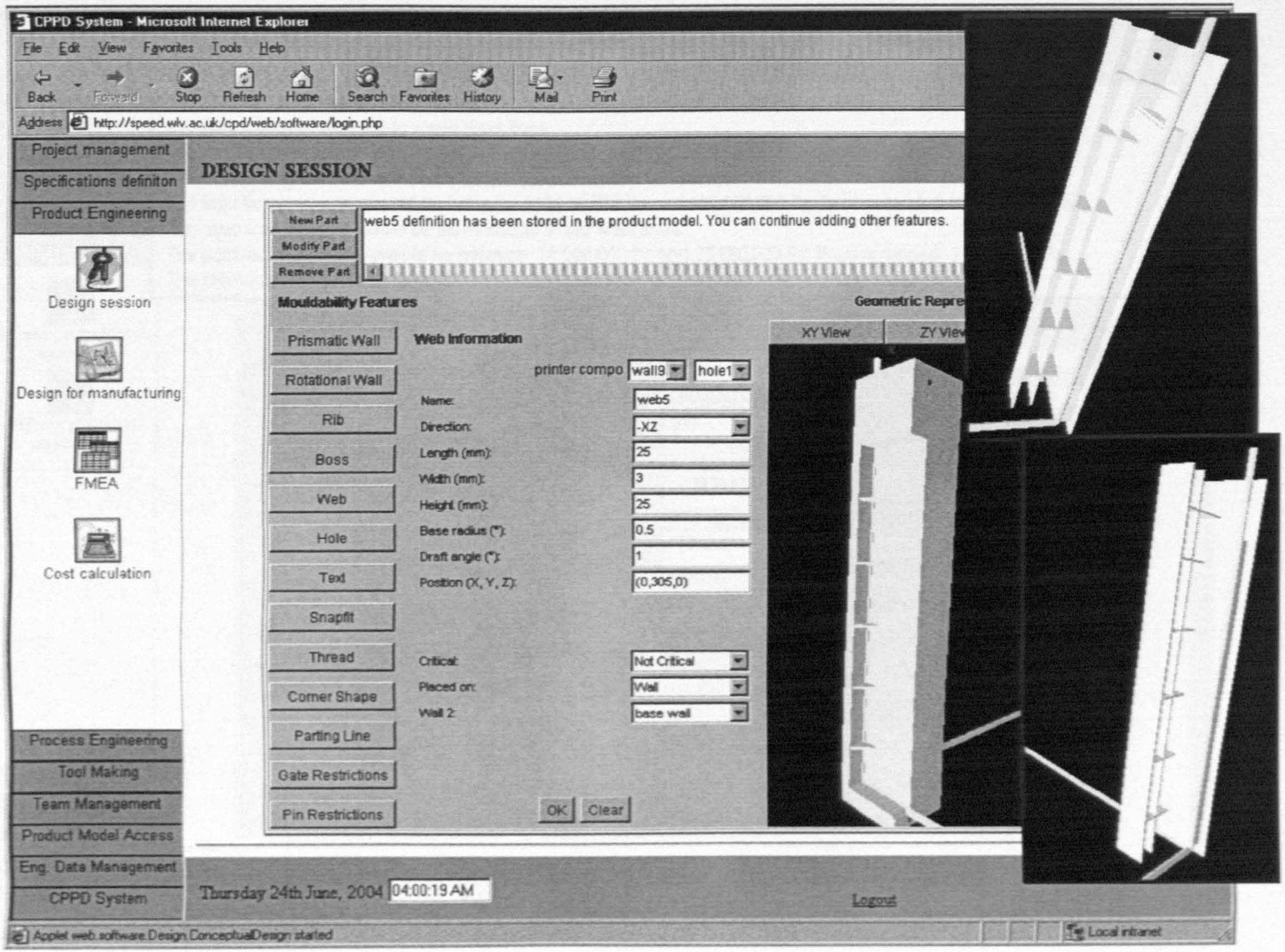


Figure 11.15 "Printer's component" plastic part design definition stored through the Design Session application

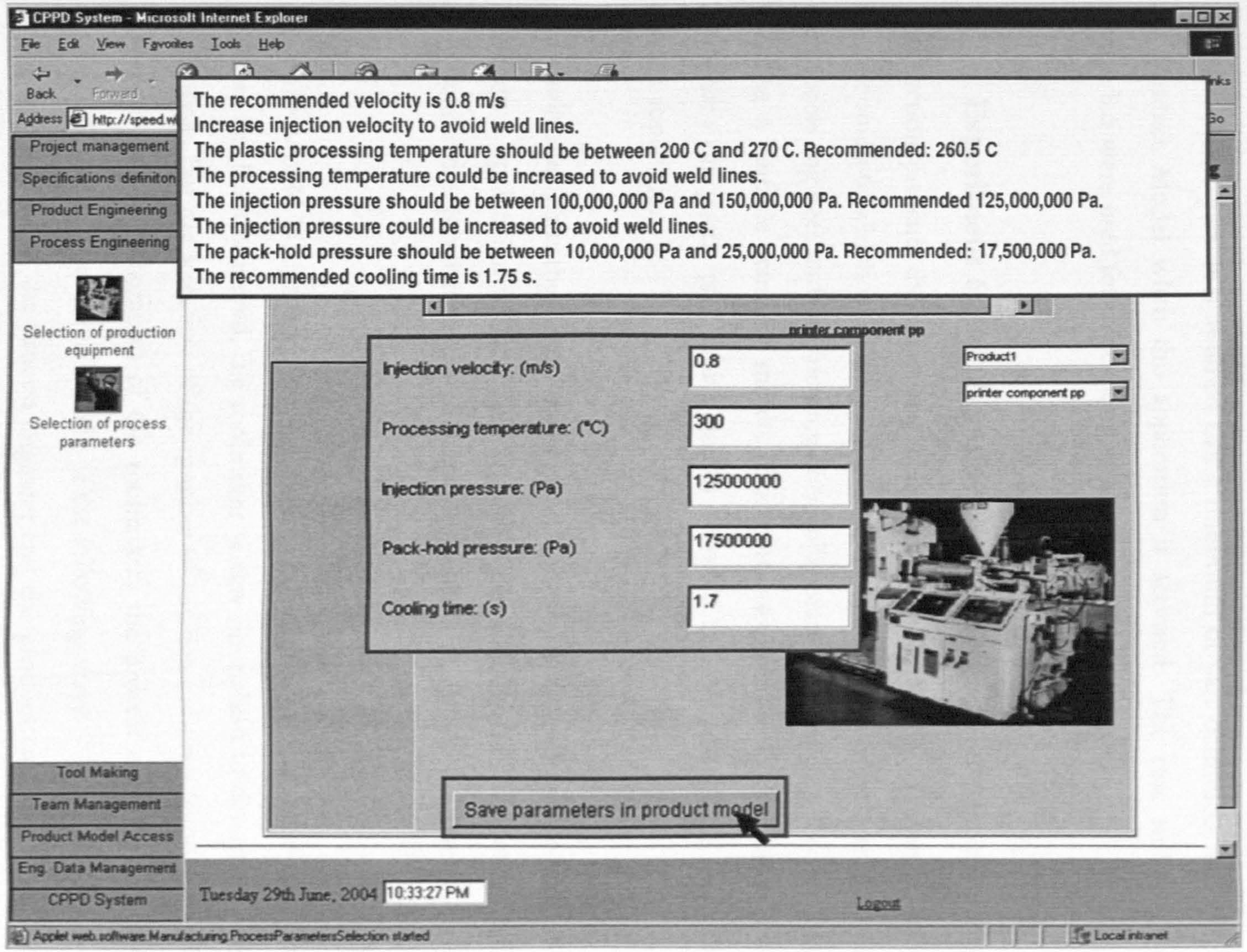


Figure 11.16 Feedback advice provided by the Selection of Process Parameters application regarding the suitable process parameters to use for production

11.5 Experimentation of the Mould Design and Fabrication application

The toolmaker uses this application to design an injection mould for the previously designed plastic part. It is assumed that a minimum of part definition has been stored in the Product Model when this application is accessed. The case studies presented in section 8.5 were used for the experimentation of this application.

11.5.1 Experiment 6: collaborative design of the injection mould plates

This section presents the experimentation done in the KdCPD system prototype using the case study 6, which was presented in section 8.5.1. In the experiment, the toolmaker, the process engineer and product engineer collaborate for designing the mould plates and selecting a suitable standard mould. The part used for this experiment is the liquid container's cap plastic part, which data is stored in the Product Model using the Design Session application.

The toolmaker starts the design of the mould plates by accessing the Mould Design and Fabrication application in the KdCPD system. The design of the plates can be started even when the product engineer has not finished to define the part in the Product Model.

Once the Mould Design and Fabrication application has been accessed and the available part data has been retrieved, the toolmaker selects the option to design an insert/bolster mould type. For the selection of the number of cavities and their layout in the mould, it is necessary the collaboration of the toolmaker, the process engineer and the product engineer. This can be achieved in any of the following ways:

- By the toolmaker, the process engineer and the product engineer using NetMeeting communication tools to decide the best number of cavities and their layout.
- By the toolmaker defining the number of cavities and their layout in the Product Model so the process engineer and the product engineer can access it.

- By the application retrieving this data from the Product Model in case the injection machine has been selected. In this case, the number of cavities and layout has already been defined.

The number of cavities and their layout are selected in the input area of the Mould Design and Fabrication application (see figure 11.17). Following this, the toolmaker clicks on the button "Recalculate Standard Mould" in order to request the system the calculation of the plate's dimensions and the selection of a suitable standard mould. This is automatically done by sharing and providing product life cycle knowledge stored in the Manufacturing Knowledge Model. The selected standard mould plate and its dimensions are then displayed by the application in the input area (see figure 11.17).

The toolmaker then clicks the "OK" button in the input area in order to store the data of the plate's dimensions in the Product Model.

11.5.2 Experiment 7: collaborative design of the gating, ejection and venting system of the injection mould

This section presents the experimentation done in the KdCPD system prototype using case study 7, which was presented in section 8.5.2. The batteries' cover plastic part is used to experiment the collaboration between the toolmaker and product engineer during the following scenarios:

- The consideration of the gating position and type
- The consideration of the ejection position and type
- The consideration of the vents position

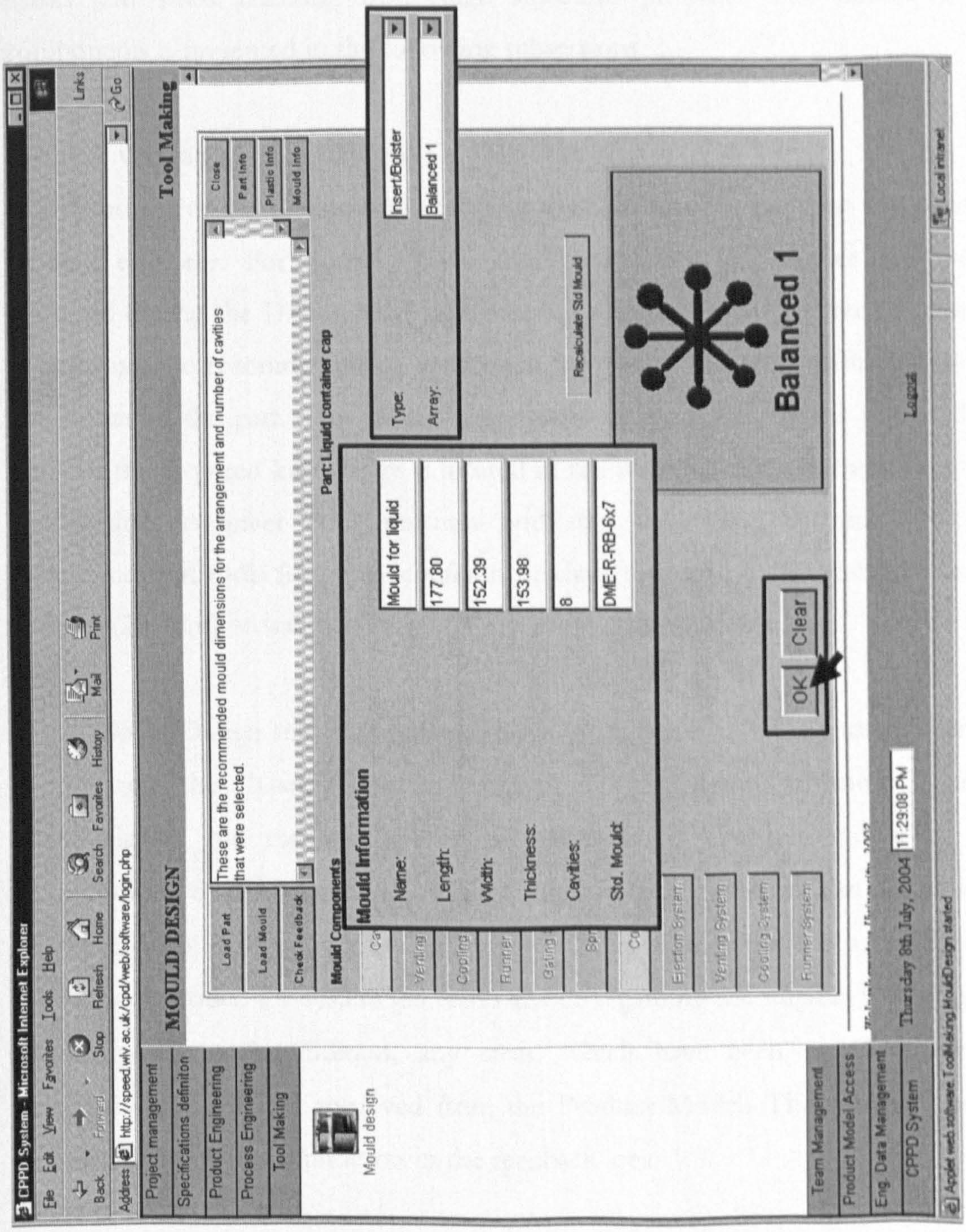


Figure 11.17 Feedback advice provided by the Mould Design and Fabrication application regarding the suitable standard mould for the "Liquid container cap" plastic part

Initially, the product engineer defines the plastic part in the Product Model using the Design Session application. Afterwards, the toolmaker accesses the Mould Design and Fabrication application in order to design the gating, ejection and venting system in the mould. These components are designed based on the feedback advice produced by the Manufacturing Knowledge Model in order to avoid short shots, weak welding lines, burn

marks and stress resulting from high injection pressure. The definition of these components is presented in the following subsections.

11.5.2.1 Gating system design

The definition of the gate positions requires the collaboration between the toolmaker and product engineer. For this, feedback advice regarding the suitable gate positions is delivered during the Design for Manufacturing application, as presented in case study 3. In this case, the recommendation is to place the gate in the point farthest away from the two bosses in the part. This advice is produced transparently by the system despite the fact that the required knowledge is located in the tool making site. Based on this advice, the product engineer communicates with the toolmaker by using Net Meeting communication tools (i.e. videoconference, chat) to define a candidate gate area in the position (25,40,0) of wall3.

In the Mould Design and Fabrication application, the design of the gating system starts by clicking on the "Gating System" button in the menu of mould components. Automatically, the manufacturability of the part is checked and as it is within manufacturing constraints, the knowledge to support the gating system design is invoked. As explained in case study 7 (see section 8.5.2), after querying the part definition from the Product Model, the system generates advice regarding the suitable type of gate and its suitable positions. In addition, any areas, which have been stored as candidate or restricted areas, are also retrieved from the Product Model. The feedback produced is then displayed by the application in the feedback area:

The suitable types of gates are:

- *Edge gate*

Following this advice, the toolmaker selects an edge gate from the choice of gates in the input area (see figure 11.18). The manufacturing constraints for the edge gate are then invoked and as described in case study 7, the feedback shown in figure 11.18 is produced. Based on this feedback, the toolmaker inputs the data of the edge gate in the input area and stores it in the Product Model by pressing the "OK" button (see figure 11.18).

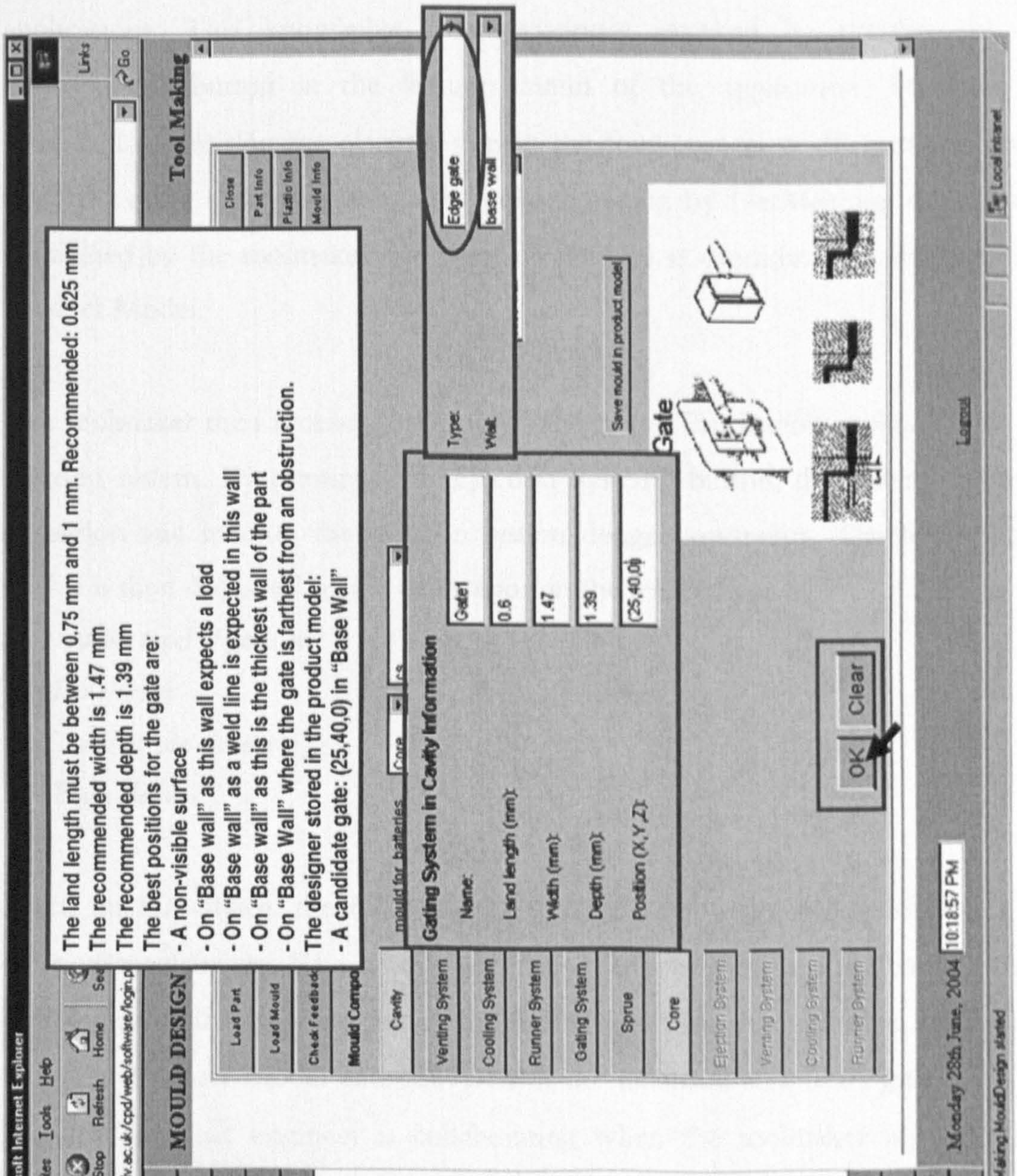


Figure 11.18 Gating system design definition application stored through the Mould Design and Fabrication

11.5.2.2 Ejection system design

The ejection system, specially the ejection positions, should be carefully considered by both the product engineer and toolmaker if undesired marks and problems with the part want to be avoided. The collaboration is supported by providing the knowledge

regarding the suitable ejection positions through the Design for Manufacturing applications. This knowledge is transparently invoked by pressing the "Ejection Restrictions" button in the features menu of the application. Feedback advice is produced to position the ejection pins in the four corners of the part and the complex boss. By using the communication tools providing by NetMeeting, these positions are confirmed by the toolmaker and they are defined as candidate ejection positions in the Product Model.

The toolmaker then accesses the Mould Design and Fabrication application to design the ejection system. By pressing the "Ejection System" button, the system queries the part definition and invokes the ejection system design constraints. The following feedback advice is then displayed by the application in the feedback area:

The suitable types of gates are:

- *Pin ejection*
- *Two step pin ejection*
- *Blade ejection*

Based on the advice, the toolmaker selects the "Pin ejection" option from the list of ejection mechanisms. The system generates advice regarding the suitable positions and dimensions of the pins. As part of the feedback advice, the restricted areas are retrieved from the Product Model and delivered in the feedback area (see figure 11.19). In such way, the product engineer is collaborating when the toolmaker is in the process of designing the ejection system. Following this advice, the toolmaker inputs the data for each of the ejector pins in the input area and stores it in the Product Model by pressing the "OK" button (see figure 11.19).

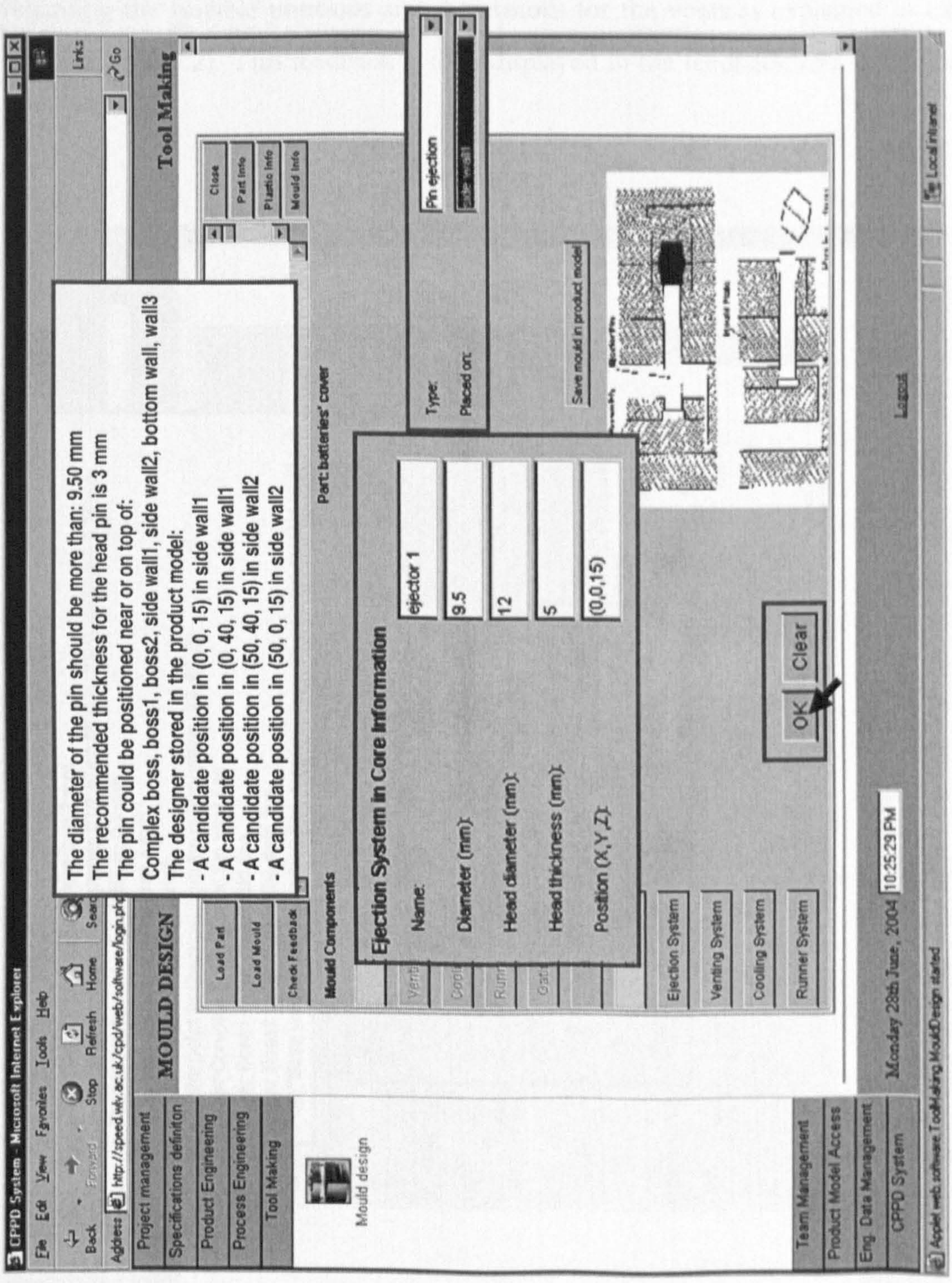


Figure 11.19 Ejection system design definition stored through the Mould Design and Fabrication application

11.5.2.3 Venting system design

Once the parting line, gating and ejection systems data is stored in the Product Model, the toolmaker starts designing the venting system by pressing the "Venting System" button in the mould components menu. The system then produces feedback advice

regarding the suitable positions and dimensions for the vents as explained in case study 7 (see section 8.5.2). This feedback is then displayed in the feedback area (see figure 11.20).

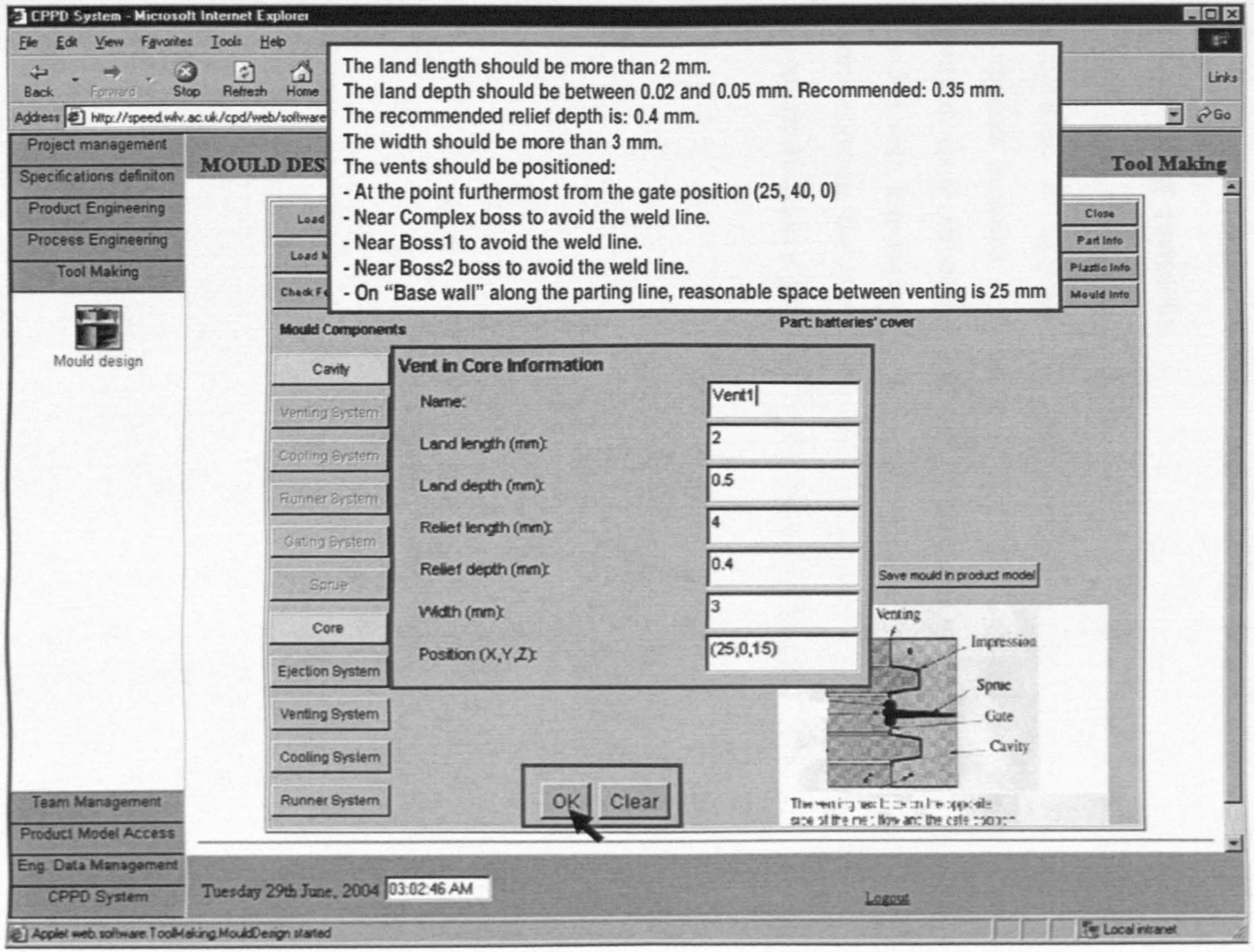


Figure 11.20 Feedback advice provided by the Mould Design and Fabrication application regarding the design of the venting system

Following the suggested values, the toolmaker inputs the data for each of the vents in the input area and stores it in the Product Model by pressing the "OK" button (see figure 11.20). The data is stored after verifying that the vent has been designed within limitations.

11.6 Closing remarks

In this chapter seven case studies were done in order to demonstrate how the collaboration is supported by the KdCPD system architecture proposed. These experiments included cases where collaboration was supported during an engineering activity to more elaborated cases where geographically distributed team members were provided with knowledge to support activities performed in different locations. For the experimentation, the case studies described in chapter 8 were used. Further discussion of the experimentation is presented in the following chapter.

Chapter 12

Discussion of Experimental Results

12.1 Introduction

The goal of the experimentation presented in the previous chapter was to demonstrate how the proposed KdCPD system architecture effectively supports collaborative product development by providing product life cycle knowledge and data in the time, place and format required. Chapter 9 and 10 presented the object oriented design and implementation of such prototype. This chapter will summarise and discuss the findings of the experimentation.

12.2 Discussion of the experimental results regarding the information layer

12.2.1 Manufacturing Knowledge Model

As discussed in section 2.3.6, many authors consider product data as knowledge which could support product development. However, the experiments have demonstrated that in addition to a Product Model, there is a need for a Manufacturing Knowledge Model in order to share and provide product life cycle knowledge of the geographically distributed companies. This model, which was presented in chapter 7, structures the manufacturing constraints that limit engineering decisions during the activities of product development. This finding supports the hypothesis that collaborative product development could be effectively supported by a system architecture that captures and provides manufacturing constraints to support engineering decision making.

Moreover, it was clear that capturing product life cycle knowledge in a Manufacturing Knowledge Model and data in a separate Product Model is an adequate approach. As

presented in experiment 1, the part definition is firstly stored in the Product Model during the Design Session application and the knowledge is thereafter applied on the data to produce feedback advice during the Design for Manufacturing application. The independence of these models allows their maintenance, update or replacement without affecting the overall structure of the KdCPD system architecture.

It was also evident that the integration of manufacturing constraints to other engineering data such as plastic material, or standard mould parts data further supports decision making. For example, in experiment 6, the data of standard mould parts from a mould provider is used to determine suitable standard mould plates.

The experiments also proved how the knowledge integrity of the Manufacturing Knowledge Model ensures the consideration of product life cycle knowledge at every stage of product development even when a specific company does not have the required knowledge. This is because the knowledge of each partner is captured and their interactions are represented in the model, as explained in section 7.5.1. This was demonstrated in experiment 2. During this experiment the knowledge of the product engineer is shared with the toolmaker in order to ensure the manufacturability of the part before starting the mould design.

12.2.2 Product Model

The findings from the literature survey and field study highlighted the need to have a Product Model structure that captures data from different activities of product development. The experiments proved this finding and demonstrated how a common source of product life cycle data could support decision making during collaborative product development. A straightforward example was shown in experiment 4, where machine data is stored in the Product Model and shared with the toolmaker, who uses this data for the calculation of suitable mould plates. The availability of a Product Model helps to address the problem of sharing data among the geographically distributed extended enterprise. This problem was identified during the activity modelling in chapter 5 (see section 5.4.2).

The structure of the design data in the Product Model was based on a feature based approach. Experiment 1 demonstrated that this approach successfully supports the provision of product life cycle knowledge. In this experiment, knowledge regarding each feature of the plastic part is applied on the design data and feedback advice regarding the problems with the features is produced. Nevertheless, the experiments highlighted that the design data structure can only represent a range of symmetrical prismatic and rotational parts and, therefore, the representation of more complex parts is restricted. Despite this, it should be stressed that the validity of the knowledge is not affected by the shape of the part but by its data. This was confirmed through experiment 4. As illustrated in figure 11.13, the design data in this experiment cannot be completely represented with the set of features provided by the Product Model. For this, features such as irregular walls are required. However, the design considerations for a wall are the same regardless the shape of the wall. Hence, it can be stated that a complex feature can be represented as a simple feature and the product life cycle knowledge is still valid.

12.3 Discussion of the experimental results regarding the application layer

The experimentation has demonstrated that the activities, supported by the KdCPD system, are not only design activities, but a range of activities that need to be performed in collaboration as identified in the activity modelling (see chapter 5). These are: design session, design for manufacturing, selection of production equipment, selection of process parameters and mould design and fabrication.

Furthermore, the results of the experiments established that the distance product development is supported by different mechanisms:

- By providing and sharing in real time both product life cycle knowledge and data, facilitating the following:
    - One engineer interacting with the system, while the other team members are able to observe and trace the product development by accessing the results. This was demonstrated in experiment 1, where the product engineer ensures the

manufacturability of the part while the process engineer can trace and access the design data in the Product Model.

- Two or more engineers, such as the product engineer and process engineer, are able to use different engineering applications simultaneously to develop a product. Experiment 2 illustrated how toolmaker and product engineers collaborate to ensure the manufacturability of the part while using the Mould Design and the Design for Manufacturing applications.
- One or more engineers, who do not have the required knowledge, are able to perform an engineering application following the advice provided by the system. This was demonstrated in experiment 3 and 7, where the product engineer has access to the knowledge of the toolmaker regarding the suitable gate and ejection positions in the part. Hence, the product engineer is able to take action during the design activity regarding preferred or not preferred areas for having gates or ejections marks.
- One or more engineers are able to access in real time the feedback advice produced by the system during the engineering applications. Experiment 1 described how the feedback advice regarding the manufacturability of the part is stored in a file to be shared amongst the geographical distributed team. This could help to have a level of traceability of the changes that have been done to the part.

- By including a communication tool, such as NetMeeting™ (Microsoft Corporation 1996-2000), to provide an environment where the geographically distributed team members can meet to perform activities in collaboration. For this purpose, a NetMeeting session can be started during the engineering activities that require collaboration of the geographically distributed team members, such as "Design Session" and "Design for Manufacturing".

Furthermore, the experimentation with the Design Session engineering applications also proved that the technology used to develop the 3D virtual environment was satisfactory in order to represent a virtual geometric model based on product data captured in the Product Model. Figures 11.5, 11.9, 11.10, 11.13 and 11.15 illustrate the virtual geometric

model of the conducted experiments. As shown in the figures, the geometry of the features is somehow sketchy. This is because in order to generate a complex and detailed geometry equal to the one provided by the solid modelling tools many man hours are required to build a complete library of geometries. In spite of this, the 3D virtual environment built for this prototype still proves how 3D technologies could be integrated to a common source of product life cycle knowledge and data which is shared and provided when taking decisions. This issue was highlighted during the mapping of the literature survey and the field study. As presented in the literature survey in chapter 2, there is no current research which achieves these results. Therefore, the decision to develop the Product Model independent from the Manufacturing Knowledge Model in the KdCPD system architecture is further supported. This will allow the extension or replacement of the data structure at any time to fit a new 3D virtual environment, which can still make use of the knowledge.

12.4 Discussion of the experimental results regarding the end user layer

The results of the experiments suggested that having a web based interface supports an easy access of the product life cycle data and knowledge through the engineering applications. As such, the KdCPD system does not need to be installed in every computer and it is only required to have access to an Internet browser. This enables the geographically distributed engineers to access to the system when they are at work, at home or in a business trip. The graphical user interface was designed using a simple layout. The different areas of the interface are clearly identified with titles and a feedback window is included in order to guide the user at every stage during the different activities that the application supports.

12.5 Other issues for discussion

Throughout the development of the system there were opportunities to present this research to several manufacturing industries, including OEMs, toolmakers and plastic

processors. Their feedback has confirmed the potential of the proposed KdCPD system prototype for supporting a real industrial engineering environment. Even more, some of the following issues have been highlighted as possible enhancements of the prototype:

- To be able to import the design from a CAD system, instead of rewriting it during the Design Session application. The author recognises the importance of this issue. To achieve this, collaboration with a commercial CAD software house will be required in order to gain access to the source code of these systems.
- To export the design geometry to a CAD system after the manufacturability of the part has been ensured. As the above point, the linkage to these systems depends of interfacing the KdCPD system with current commercial software applications.
- To automatically modify the product data in the Product Model, when accepting the suggestions produced by the KdCPD system regarding the manufacturability of the part. This is an implementation issue, which can be customised when implementing a system for particular extended enterprise.
- To include more knowledge, such as machining constraints or cost calculations constraints, in the Manufacturing Knowledge Model. The issue of extending the knowledge will be addressed as further work. For the purpose of this research, the knowledge which was captured in the system prototype clearly demonstrated the advantage of having such model.

In an overall analysis, the proposed KdCPD does not aim to replace existing systems in the companies but rather to be a support tool for communicating and sharing knowledge among the geographically distributed partners. As such, the implementation of this system could be considered feasible among the partners of one industrial group or extended enterprise, who are bonded by common financial interests.

Chapter 13

Conclusions and Further Work

13.1 Research conclusions

In order to remain competitive in a global market, companies have adopted a geographically distributed working approach. As such, activities like design and manufacturing are being performed by companies located in different places and countries. These companies need to collaborate effectively in order to develop their product faster, more cost effective and with better quality. This global engineering environment has led to the distribution of product life cycle information and knowledge affecting the collaboration throughout product development. The sources of this knowledge are the experience of individuals, corporate information of past products and projects, published literature, as well as the manufacturing process and resource capabilities. Collectively, this knowledge constrains the engineering decisions that could be made during product development. It is critical to provide this knowledge in the right time, place and format to support engineering decision making throughout product development within the extended enterprise. In order to address this need, the research focused primarily in proposing a knowledge driven system architecture for collaborative product development based on manufacturing constraints.

The approach taken in this research for proposing such system architecture was to adopt a practical methodology based on both findings from extensive analysis of existing CPD systems and an industrial field study. This methodology was useful to identify the evolving and emerging research issues and based on this to develop an adequate solution. The conclusions, which can be drawn from this research work, are the following:

1. The literature survey, presented in chapter 2, has shown that the research and commercial development undertaken in CPD systems have focused on providing

Internet based applications to share design data, such as product geometry, between distributed users. However, the capture, representation and provision of product life cycle knowledge in a collaborative environment have not been addressed by the research community or the commercial applications.

2. Chapter 3 presented a field study which clearly demonstrated the industrial requirement to have a solution that is able to support collaborative product development by sharing product life cycle data and knowledge. Mapping this requirement with the findings of the reviewed systems led to the identification of the key research issues of this work.

3. The development of the KdCPD system architecture was based on a widely accepted reference framework. CIMOSA has proven to be a suitable reference framework to develop the architecture by modelling the different aspects of the enterprise, such as activities, information and organisation. However, the reference architecture had to be adapted to suit the requirements of the research. The location aspect was added because it was necessary to represent the location of the distributed information and knowledge.

4. During the activity modelling presented in chapter 5, IDEF0 proved to be an understandable and easy to use modelling technique to identify the information and knowledge driven manufacturing activities. However, an enrichment was needed in order to represent the way in which the activities are done in different locations. For this, an additional location notation was introduced to represent the different sites where activities are developed in the extended enterprise.

5. The components of the proposed system architecture KdCPD were selected in accordance with the findings of the previous stages of the research. Furthermore, the design and implementation of such system architecture (see chapter 9 and 10) helped to prove how it addresses the evolving and emerging research issues. This was done through the experimentation of several case studies (chapter 8 and 11) in the

implemented system prototype. The findings of these experiments demonstrated that the proposed system architecture effectively addresses the need to capture and provide in real time product life cycle knowledge in order to support collaborative product development among the extended enterprise. Furthermore, the experiments demonstrated how the knowledge integrity and geographical distribution of the Manufacturing Knowledge Model can ensure that the right decisions are taken during the different stages of product development.

6. The knowledge which belongs to an extended enterprise was identified, captured and formally represented in a Manufacturing Knowledge Model. Chapter 7 presented this knowledge modelling process in detail. The integration of this knowledge was achieved by using the enabling technology UML to formally represent the manufacturing constraints and their interactions. UML notation provided a clear and standard representation, which was transparently used during the design of the KdCPD system. UML was also used for designing the system architecture. This provided the advantage of reusing the different models done throughout the research and of having a standard representation of information, knowledge and applications that composed the system architecture. The location notation was used throughout all these models in order to represent the different sites of an extended enterprise.

7. Internet technologies, such as Java, Java 3D, Corba and PHP were chosen as enabling technologies to implement the KdCPD system prototype. These technologies proved to be suitable for developing Internet applications. Even more, they are widely supported, as they are free of cost and popular among the research community. The use of an object oriented database manager provided the robustness required from the prototype by capturing the complexity of the product life cycle knowledge and data in a suitable format. The integration of all these technologies achieved in this research provided a reusable and reliable KdCPD prototype implementation.

8. NetMeeting™ was selected to support real time communication among the geographically distributed engineers because it was considered to be a tool, which is

widely available, easy to use and free of cost. These characteristics facilitate the adoption of this technology to enhance the use of the decision support engineering applications provided by the KdCPD system. Nevertheless, it is important to mention that other tools that also support real time communication and that provide the same advantages as NetMeeting™ could also be considered in future work.

9. The proposed KdCPD system supports the geographically distributed team members throughout product development providing the following advantages:
   - Improves the effective communication and collaboration of the different partners when performing geographically distributed manufacturing activities.
   - Standardises the knowledge of each of the companies in a formal repository.
   - Brings together the geographically distributed knowledge of the companies while keeping their intellectual property secure.
   - Improves the sharing and provision of data and knowledge within the company and the supply chain
   - Ensures that each of the activities of product development is carried out according to the limitations of the process, resources and material.
   - Reduces product development time by eliminating most feedbacks and iterations caused by not sharing knowledge and information.

13.2 Original contributions

Chapter 2 and 3 introduced one central research issue concerning CPD systems, which has not been successfully addressed neither by the research nor the commercial community. This issue is the capture, representation and provision of knowledge related to different activities of product development in order to support decision making in a collaborative environment.

This research addresses this issue by two main original contributions. These will be described in the following subsections

13.2.1 Knowledge based system architecture

The aim of this research and therefore, one of its main contributions, was the development of a knowledge driven system architecture that supports collaborative product development by providing knowledge through engineering applications in a collaborative environment.

The development of such system architecture was guided by a methodology which enabled the representation of the different aspects of an extended enterprise. The use of different enabling technologies, such as IDEF0 and UML, to represent these aspects in a geographically distributed setting is an additional methodological contribution of the research.

13.2.2 Manufacturing Knowledge Model

The second original contribution is the Manufacturing Knowledge Model, which is a source of knowledge that captures and represents the geographically distributed product life cycle knowledge and which is the basis of the system architecture. The knowledge modelling process proposed in this research identifies, captures, and formally represents manufacturing constraints which belong to geographically distributed companies. This modelling process is also an additional methodological contribution of the research.

Chapter 4 introduced five central questions regarding the Manufacturing Knowledge Model, which was developed in this research. The questions are:

1 What knowledge must be included in the Manufacturing Knowledge Model to support decision making during CPD?
2 Where is the knowledge located?
3 How could manufacturing constraints be captured?
4 How could the geographically distributed manufacturing constraints be represented in the Manufacturing Knowledge Model?
5 How could the Manufacturing Knowledge Model provide a common source of integrated knowledge to support CPD?

The first question was answered in chapter 7, where the knowledge of key activities of product development was identified. These activities were selected according to the activity modelling performed in chapter 5 as part of CIMOSA reference framework. As such, the knowledge related to the following activities was identified: design for manufacturing, selection of production equipment, selection of process parameters, mould design and fabrication. This knowledge was located in documents, books, reports and engineer's expertise which was geographically distributed among the extended enterprise. Capturing and representing the geographic distribution of the knowledge among the companies of the extended enterprise is an additional contribution of the research. Hence, by using a Manufacturing Knowledge Model, every company in an extended enterprise is able to secure their intellectual property while still sharing this knowledge to support the decisions made by their collaborators.

The knowledge modelling process (see chapter 7) described in detail how the geographically distributed manufacturing constraints are captured and represented in the Manufacturing Knowledge Model. For this, the knowledge is captured and standardised in the form of rules and mathematic formulas, which are later formally represented using UML enabling technique.

Finally, the experimentation conducted with the KdCPD system prototype (see chapter 11) demonstrated how the Manufacturing Knowledge Model provides a common source of integrated knowledge for supporting geographically distributed engineers to make decisions during collaborative product development. Furthermore, the knowledge is represented in such a way that the impact of one manufacturing constraint on other engineering activities is highlighted. This can only be achieved by capturing the relationship between the manufacturing constraints of the different activities of the product life cycle in despite the location of the knowledge or who performs these activities. The knowledge integrity not only supports decision taking but also enables the sharing of knowledge among collaborators and ensures that correct engineering decisions are taken at every stage of product development.

13.3 Directions for further work

The author believes that there are some areas where further research could be performed in order to enhance the support provided by the KdCPD system architecture. Some of these research directions are described in the following subsections.

13.3.1 Extension of the Manufacturing Knowledge Model

In this research, the injection moulding product development knowledge was addressed; however, the knowledge modelling process presented in chapter 7 could be used to further expand the Manufacturing Knowledge Model. For example, knowledge to support activities such as cost modelling (Abdalla and Shehab 2002, Roy and Palacio 2000) could be further explored in order to enhance the capability of the KdCPD system to support more activities during product development. For this, it is necessary to expand the knowledge model by identifying, capturing and representing in UML this knowledge.

13.3.2 Complex 3D geometric representation

The limited geometric representation of complex products in the implemented prototype could be further enhanced in order to represent more complex geometries. This issue could be addressed in several ways:

- By enhancing the current 3D virtual geometric model of the system prototype: this requires of additional implementation effort to the current system prototype.
- By providing an interface with a commercial solid model: this requires an interface between the current system and a commercial CAD system. For this, the collaboration between researcher and commercial houses it is necessary in order to gain access to the code of such software.

13.3.3 Standardised Product Model

One enhancement of the KdCPD system will be the use of a standard representation of the product data, such as STEP standard (ISO 10303). STEP is a standard, which can

provide interoperability between the KdCPD system and other systems. This will provide the system with a data exchange mechanism. It can, therefore, add the functionality of importing a solid model into the system or exporting the virtual Product Model to a data exchange format or a specific commercial CAD format. This functionality will provide the KdCPD system with some interoperability with the systems already existent in the company.

The data exchange mechanism can be implemented using different options:

- The first is to develop a translator in order to export the geometric representation of the part developed in the KdCPD into a format that can be read by commercial CAD/CAM/CAE systems. Another translator would be necessary in order to convert from a commercial system into the product data representation in the KdCPD system. The latter development has further consequences as it means that a feature recognition mechanism will be required. The reason is that the knowledge captured is based on features and therefore the product data needs to be decomposed in features in order to interact with the knowledge.
- Another approach is the use of a product data representation standard, such as STEP. In this case, it would be possible to exchange STEP files in and out of the system. However, the issue of recognising the product features remains, as this standard does not provide a feature based approach for storing the product data. Therefore, further manual intervention will have to be done by the user, such as the selection of the features from the geometry.

13.3.4 Further development of decision support engineering applications

The developed prototype shows how some of the applications of the architecture could be implemented. However, other elements of the architecture remain to be fully explored. Some of the applications, such as project management, require further research in order to implement a more complete solution.

13.3.5 Consideration of the human factor involved in CPD

The study of the engineer's behaviour during the collaboration using the proposed KdCPD system architecture needs to be considered. Human related problems could arise from the engineer to engineer interaction as well as from the fact that the engineers reside in different countries; hence having different cultures, languages and timetables. These issues include conflict resolution, human interaction, and others.

13.3.6 Commercial exploitation of the KdCPD system architecture

As mentioned in the discussion of the experimentation, the implementation of the proposed KdCPD system is considered feasible among an extended enterprise which is bounded by common commercial interests. For doing this, it will firstly be necessary to customise the different components of the system architecture to the needs of the particular extended enterprise. This includes the customisation of knowledge, data and engineering applications.

The implementation of this system architecture in an extended enterprise further facilitates the sharing of costs of such technology. The hardware required are mainly PC based servers where the system and the databases are geographically distributed among the collaborators. The software required for the design and implementation done during this research is mostly free of cost, apart from the OODBMS development licence.

In such a way, an extended enterprise will benefit from the research performed by the author that will, in turn, support the companies to be more productive and develop products in less time and with better quality.

References

Abdalla, H., and Shehab E. (2002), "An Intelligent Knowledge-based System for Product Cost Modelling," International Journal of Advanced Manufacturing Technology, 19:49-65, 2002

Abrahamson, S., D. Wallace, N. Senin and P. Sferro (2000). "Integrated Design in a Service Marketplace." Computer Aided Design 32: 97-107.

Agile Software (2003) Agile, http://www.agile.com/ 2003

Al-Ashaab, A. (1994). A Manufacturing Model to Capture Injection Moulding Process Capabilities to Support Design for Manufacture. Loughborough, Loughborough University of Technology.

Al-Ashaab, A. and R. Young (1995). "Design for Injection Moulding in a Manufacturing Model Environment." Journal of Design and Manufacturing 5: 45-54.

Alibre Inc. (2003) Alibre, http://www.alibre.com 2003

Alventive Inc (2003) Alventive, http://www.alventive.com/ 2003

Anderson, N. and H. Abdalla (2001). Web Browser Based Collaborative Product Development for the 21st Century. International Conference on Concurrent Engineering: Research and Applications, California, USA.

Automation Creations, I. (2004) Matweb™ Materials Database, http://www.matweb.com/reference/terms.asp 2004

Bauer, M., H. J. Eikerling, W. Mueller, A. Pawlak, K. Siekierska, D. Soderberg and X. Warzee (2001). Advanced Infrastructure for Pan-European Collaborative Engineering. E-work and E-commerce: Novel Solutions and Practices for a Global Networked Economy, Venice, Italy, Brian Stanford-Smith & Enrica Chiozza.

Bellinger, G. (2003) Modeling & Simulation, an Introduction, http://www.systems-thinking.org/modsim/modsim.htm 2003

Biennier, F. and J. Favrel (1999). Collaborative Work Organisation in a Pool of Enterprises. Advances in Concurrent Engineer Conference, Bath, UK.

Booch, Grady (1994) Object-Oriented Analysis and Design with Applications, Benjamin/Cummings, second edition.

Bramall, D., K. McKay and P. Maropoulos (2001). Supporting Aggregate Process Planning with Product Process and Resource Knowledge. European Concurrent Engineering Conference, Valencia, Spain.

Browne, J., I. Hunt and J. Zhang (1997). The Extended Enterprise, Handbook of Life Cycle Engineering Concepts, Tools and Techniques.

Caldwell, N. H. M., P. J. Clarkson, P. A. Rodgers and A. P. Huxor (2000). Web-Based Knowledge Management for Distributed Design. IEEE Intelligent Systems: 40-47.

Centric Software (2003) Centric Software's Collaborative Product Innovation (Cpi), http://www.centricsoftware.com/ 2003

Chang, H.-C., W. F. Lu and X. F. Liu (1999). "Www-Based Collaborative System for Integrated Design and Manufacturing." Concurrent Engineering: Research and Applications 7(4): 319-334.

Chen, Y.-M. and M.-W. Liang (2000). "Design and Implementation of a Collaborative Engineering Information System for Allied Concurrent Engineering." International Journal Computer Integrated Manufacturing 13(1): 11-30.

Choo, W. C., B. Detlor and D. Turnbull (2000). Web Work: Information Seeking and Knowledge Work on the World Wide Web. The Netherlands, Kluwer Academic Publishers.

Chung, J. and L. Kunwoo (2002). "A Framework of Collaborative Design Environment for Injection Molding." Computers in Industry 47: 319-337.

CIMOSA Association (1996) Cimosa, a Primer on Key Concepts, Purpose and Business Valu, http://cimosa.cnt.pl/Docs/Primer/primer5.htm 2003

Coad P. and Yourdon E. (1991) Object-Oriented Analysis, Prentice Hall.

CoCreate (2003) Cocreate, http://198.65.136.44/ 2003

Coleman D., Arnold P., Bodoff S., Dollin C., Gilchrist H., Hayes F., Jeremes P., (1994) Object-Oriented Development The Fusion Method, Prentice Hall.

Colquhoun, G. J., R. W. Baines and R. Crossley (1993). "A State of the Arte Review of Idef0." International Journal of Computer Integrated Manufacture 6(4): 252-264.

Cracknell, P. S. and R. W. Dyson (1993). Handbook of Thermoplastics Injection Mould Design. London, Blackie Academic & Professional.

Dassault Systemes (2003) Smarteam, http://www.smarteam.com/homepage.asp 2003

Dassault Systemes (2003) Enovia, http://plm.3ds.com/en/about_plm/enovia.asp 2003

Dieter, G. E. (2000). Engineering Design, McGraw-Hill International Editions, 3rd Edition

Domazet, D. S., M. C. Yan, C. F. Y. Calvin, H. P. H. Kong and A. Goh (2000). An Infrastructure for Inter-Organizational Collaborative Product Development. International Conference on System Science, Hawaii, IEEE.

Dow (2001). "Product and Mold Design."

e2open (2003) E2open, http://www.e2open.com/ 2003

EDS (2003) Teamcenter, http://www.eds.com/products/plm/teamcenter/products.shtml 2003

EDS (2003) E-Vis, http://www.eds.com/products/plm/teamcenter/evis/index.shtml 2003

ESPRIT Consortium AMICE (1993). 2nd Revised and Extended Edition, Research Report, Esprit Project 688/5288, Springer-Verlag.

Eurostep Commercial Solutions AB (2003) Share-a-Space, http://www.share-a-space.com/ 2003

Fikes, R. and A. Farquhar (1999). Large-Scale Repositories of Highly Expressive Reusable Knowledge. IEEE Intelligent Systems. 14.

Fliedner, G. (2003). "Cpfr: An Emerging Supply Chain Tool." Industrial Management & Data Systems 103(1): 14 - 21.

GE Plastics (1999). Ge Engineering Thermoplastics Design Guide.

Gomez-Perez, A. and R. Benjamins (1999). Overview of Knowledge Sharing and Reuse Components: Ontologies and Problem-Solving Methods. IJCAI-99 Workshop on Ontologies and Problem-Solving Methods(KRR5), Stockholm, Sweden.

Gruber, T. R., J. M. Tenenbaum and J. C. Weber (1992). Toward a Knowledge Medium for Collaborative Product Development. International Conference on Artificial Intelligence in Design, Pittsburgh USA, Kluwer Academic Publishers.

Gupta, S. K., C. J. J. Paredis, R. Sinha, C.-H. Wang and P. F. Brown (1998). An Intelligent Environment for Simulating Mechanical Assembly Operations. ASME Design for Manufacturing Conference, Atlanta, USA, ASME.

Harrison, P. J. and M. Gulli (1996). Virtual Collaborative Simulation Environment for Integrated Product and Process Development. International Symposium on High Performance Distributed Computing, Syracuse, NY, USA.

Höfling, B. (1999). Reuse of Step Product Data for the Generation of Technical Documentation. International Joint Conference on Artificial Intelligence, Stockholm, Sweden.

Huang, G. Q. and K. L. Mak (2001). "Web-Integrated Manufacturing: Recent Developments and Emerging Issues." International Journal of Computer Integrated Manufacturing 14(1): 3-13.

Huang, G. Q., J. Shi and K. L. Mak (2000). Synchronized "Design for X" Explorer on the World Wide Web. Advances in Concurrent Engineer, Lyon, France.

Informative Graphics Corp. (2003) Brava! ,http://www.bravawebkit.com/ 2003

Integrated Development Enterprise (2003) Ide, http://www.ide.com 2003

International Comittee for Information Technology Standards (2003) T3 - Open Distributed Processing (Odp),http://www.ncits.org/tc_home/t3.htm 2003

ISL Michigan State University (1999) Injection Moulding, http://islnotes.cps.msu.edu/trp/inj/geo_weld.html 2003

Jacobson, Ivar et al. (1992) Object-Oriented Software Engineering. Addison-Wesley.

Jae, L. Y., K. Hyun and K. Kwangsoo (2001). "A Web-Enabled Approach to Feature-Based Modeling in a Distributed and Collaborative Design Environment." Concurrent Engineering: Research and Applications 9(1).

Kifer, M., G. Lausen and J. Wu (1996). Logical Foundations of Object-Oriented and Frame-Based Languages. 9th European Knowledge Acquisition Workshop, EKAW-96, Springer-Verlag.

Kim, C.-Y., N. Kim, Y. Kim, S.-H. Kang and P. O'Grady (1998). Internet-Based Concurrent Engineering: An Interactive 3d System with Mark-Up. ASME 18th Computter in Engineering Conference, Atlanta, USA, ASME.

Knowledge based Systems Inc. (2000). A structured approach to enterprise modelling and analysis, http://www.idef.com 2003

Knublauch, H. and T. Rose (2000). Round-Trip Engineering of Ontologies for Knowledge-Based Systems. International Conference on Software Engineering and Knowledge Engineering, Chicago, USA.

Latif, M. N. and R. G. Hannam (1996). "Feature-Based Design and the Object-Oriented Approach." Journal of Engineering Design 7(1): 27-37.

Lee, W., C. F. Cheung, H. C. W. Lau and K. L. Choy (2003). "Development of a Web-Based Enterprise Collaborative Platform for Networked Enterprises." Business Process Management Journal 9(1): 46 - 59.

Li, G., R. H. Bracewell and K. M. Wallace (2001). A Knowledge Framework to Support Engineering Design. International Conference on Concurrent Engineering: Research and Applications, California, USA.

Lu, S. C.-Y. and J. Cai (2000). Stars: A Socio-Technical Framework for Integrating Design Knowledge over the Internet. IEEE Internet Computing. September-October 2000.

Matrix One (2003) Matrix10, http://www.matrixone.com 2003

Mayer, R. J., C. P. Menzel, M. K. Painter, DeWitte, T. Blinn and B. Perakath (1995). Information Integration for Concurrent Engineering (Iice), Idef3 Process Description Capture Method Report, Knowledge Based System Inc., USA.

Menges, G., W. Michaeli and P. Mohren (2001). How to Make Injection Moulds. Munich, Hanser Publisher.

Mennig, G. (1998). Mold-Making Handbook. Munich, Hanser Publisher.

Microsoft Corporation (1996-2000) Windows Netmeeting, 3.01,

Miller, R. (2003) Practical Uml™: A Hands-on Introduction for Developers, http://bdn.borland.com/article/0,1410,31863,00.html#classdiagrams 2004

Miller, T. L. (1996) Injection Moulding, http://www.devicelink.com/mddi/archive/96/04/008.html 2004

Molina, A. (1995). A Manufacturing Model to Support Data-Driven Applications for Design and Manufacture. Loughborough, Loughborough University of Technology.

Molina, A., A. Al-Ashaab, T. Ellis and R. Young (1995). "A Review of Computer-Aided Simultaneous Engineering Systems." Research in Engineering Design 7: 38-63.

National Industrial Information Infrastructure Protocols (2001) National Industrial Information Infrastructure Protocols and Related Projects, http://www.niiip.org 2003

Netscape Network (2001-2003) Javascript Central, http://devedge.netscape.com/central/javascript/ 2004

NIST (2004) National Institute of Standards and Technology, http://www.nist.gov 2004

Object Management Group (2003) Corba Basics, http://www.omg.org/gettingstarted/corbafaq.htm 2003

Object Management Group (2003) Unified Modeling Language, http://www.uml.org 2003

Oracle (2003) Oracle Product Development Exchange, http://www.oracle.com/appsnet/products/exchanges/docs/product_development_exchange.html 2003

Osswald, T., L.-S. Turng and P. Gramann (2002). Injection Moulding Handbook. Munich, Hanser Publisher.

Owen, J. (1994). Step- an Introduction. Winchester, UK, Information Geometers Ltd.

Primavera Systems Inc (2003) Primecontract, http://www.primavera.com/products/primecontract.html 2003

Progress Software (2003) Objectstore, http://www.objectstore.net/index.ssp 2003

Project Management Institute (2000). Guide to the Project Management Body of Knowledge, a (Pmbok® Guide).

PTC (2003) Pro/Engineer Design Collaboration, http://www.ptc.com/appserver/it/icm/cda/icm01_list.jsp?group=201&show=y&keyword=868 2003

PTC (2003) Windchill Pro/Engineer Extension, http://www.ptc.com/appserver/it/icm/cda/icm01_list.jsp?group=201&num=1&show=y&keyword=982 2003

Pye, R. G. W. (1989). Injection Mould Design.

Qiang, L., Y. F. Zhang and A. Y. Nee (2001). "A Distributed and Collaborative Concurrent Product Design System through the Www/Internet." Advanced Manufacturing Technology 17: 315-322.

Qin, S. F., R. Harrison, A. A. West, I. N. Jordanov and D. K. Wright (2003). "A Framework of Web-Based Conceptual Design." Computers in Industry 50: 153-164.

Rezayat, M. (2000). "The Enterprise-Web Portal for Life-Cycle Support." Computer Aided Design 32: 85-96.

Rodgers, P., N. H. Caldwell, P. J. Clarkson and A. Huxor (2001). "The Management of Concept Design Knowledge in Modern Product Organizations." International Journal of Computer Integrated Manufacturing 14(1): 108-115.

Rodriguez, K. and A. Al-Ashaab (2002). A Review of Internet Based Collaborative Product Development Systems. International Conference on Concurrent Engineering: Research and Applications, Cranfield, UK.

Roy R. and Palacio A. (2000), "Cost Estimating and Risk Analysis in Manufacturing Processes", Proceedings of MATADOR 2000 Conference, 13-14 July, Manchester, UMIST, ISBN 1-85233-323-5, pp. 177-182, 2000.

Roy, U., B. Bharadwaj, S. S. Kodkani and M.Cargian (1997). "Product Development in a Collaborative Design Environment." Concurrent Engineering: Research and Applications 5(4).

Rumbaugh, J., I. Jacobson and G. Booch (1999). The Unified Modeling Language Reference Manual, Addison-Wesley Object Technology Series.

Schlumberger Limited (2003) Mindshare, http://www.sis.slb.com/content/software/knowledge/mindshare.asp 2003

Sehdev, K., I. Fan, S. Cooper and G. Williams (1995). Design for Manufacture in the Aerospace Extended Enterprise. World Class Design to Manufacture. 2: 28-33.

Sevy, J., V. Zaychik, T. T. Hewett and W. Regli (2000). Deploying and Evaluating Collaborative Engineering Studios. International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Gaithersburg, USA.

Shi, J., G. Q. Huang, K. L. Mak and C. H. He (2001). Web-Based "Design for X" for Collaborative Product Development. International Conference on Concurrent Engineering, California, USA.

Shyamsundar, N. and R. Gadh (2001). "Internet-Based Collaborative Product Design with Assembly Features and Virtual Design Spaces." Computer-Aided Design 33(9): 637-651.

Silicon Graphics, I. (2003) Open Inventortm 2003

Sommerville, I. (2001). Software Engineering. Essex, England.

Su, D., S. Ji, J. Li and X. Chen (2002). Web-Enabled Collaborative Environment for Integrated Design and Manufacture. International Conference on Concurrent Engineering: Research and Applications, Cranfield, UK.

Sun Microsystems (2003) The Source for Java Technology, http://java.sun.com 2003

Swartout, W., R. Patil, K. Knight and T. Russ (1997). Toward Distributed Use of Large-Scale Ontologies. Symposium Series on Ontological Engineering, California, USA, AAAI Press.

TC184/SC4 Organization (2004), STEP Overview, http://www.tc184-sc4.org/SC4_Open/SC4_Work_Products_Documents/STEP_(10303)/ 2004

TDCI Solutions (2003) TDCI, http://www.tdci.eu.com/ 2003

The Open Group (2000) Distributed Computing Environment (Dce) 2003

The PHP Group (2001-2004) PHP, http://www.php.net/ 2004

Ticona (2000). Designing with Plastic.

Ticona and Tim Spahr (2000). Snap-Fit for Assembly and Dissasembly.

Törlind, P. (1999). Product Models and Environment for Distributed Engineering. Luleå Sweeden, Luleå University.

Wang, Y. D., W. Shen and H. Ghenniwa (2003). "Webblow: A Web/Agent-Based Multidisciplinary Design Optimization Environment." Computers in Industry 52(1).

Webscope Inc (2003) Webscope, http://www.webscopeinc.com/ 2003

Young, R. and R. Bell (1992). "Machine Operation Planning in Product Development Modelling Environment." International Journal of Production Research 30(11): 2487-2513.

Young, R., S. Wang, J. M. Dorador, C. Costa and J. Zhao (2001). A Shared Information/ Knowledge Environment to Support Concurrent Engineering. International Conference on Concurrent Engineering: Research and Applications, California, USA.

Zachman, J. (1997). Concepts of the Framework for Enterprise Architecture, Zachman International.

Zaychik, V., W. C. Regli and T. T. Hewett (2000). Evaluation-Guided Research and Development for Collaborative Engineering Environments. Conference for the Computer-Human Interaction Special Interest Group of the Ergonomics Society of Australia, Sydney, Australia.

Zaychik, V., J. Sevy and W. Regli (2000). A Collaborative Design Studio: Architecture and Prototype. CoDesigning 2000 Conference, Coventry, UK.

Zhan, H. F., W. B. Lee, C. F. Cheung, S. K. Kwok and X. J. Gu (2003). "A Web-Based Collaborative Product Design Platform for Dispersed Network Manufacturing." Journal of Materials Processing Technology 138(1-3): 600-604.

Zhuang, Y., L. Chen and R. Venter (2000). "Cybereye: An Internet-Enabled Environment to Support Collaborative Design." Concurrent Engineering: Research and Applications 8(3): 213-229.

Appendix A

Questionnaire for Field Study

THE NEED FOR COLLABORATIVE PRODUCT DEVELOPMENT SYSTEM

1. How important is it for your company to collaborate with partners and supply chain that are geographically distributed?

☐ No need. ☐ Regular ☐ Important ☐ Crucial

2. How often do you need to communicate with other engineers who are located remotely?

☐ Daily ☐ Weekly ☐ Monthly ☐ Annually ☐ Occasionally

3. How important is it to share the knowledge (how to do things) with other engineers involved in product development within the same company and the supply chain?

☐ Not important ☐ Regular ☐ Important ☐ Crucial

4. How important is it to share the product data with other engineers involved in product development within the same company and the supply chain?

☐ Not important ☐ Regular ☐ Important ☐ Crucial

5. Do you think a computer software that helps you to collaborate and share knowledge and product information through Internet will help you to improve your work?

☐ Yes ☐ No

MECHANISM FOR COLLABORATION

6. Which percentage of your time do you collaborate with engineers inside the company and with engineers in the supply chain?

_____% Inside the company _____% With the supply chain

7. With whom do you need to collaborate?

☐ Designer ☐ Suppliers

☐ Tool makers

☐ Others_____

8. Which are the preferable mechanisms of communication in the company to collaborate with the supply chain?

☐ Phone _____% ☐ Post _____%

☐ E-Mail _____% ☐ Visiting _____%

☐ Videoconferencing _____% ☐ Commercial CPC _____%

☐ Other _____

9. Do you think is necessary to use other mechanisms to collaborate using Internet?

☐ No need. ☐ Not important ☐ Regular ☐ Important ☐ Crucial

10. Do you need to collaborate using Internet in real time (visualising the geometry of the product) or interacting not in real time (transfer a file, send an email)?

☐ Not real time collaboration ☐ Both types of collaboration ☐ Real time collaboration

11. Which will be the best mechanisms to collaborate using Internet?

☐ Sharing the geometry of the design in real time
☐ Sharing product information (PDM systems)
☐ Video
☐ Other_____
☐ Email
☐ Chat Session
☐ Audio

INFORMATION AND KNOWLEDGE USED WHEN COLLABORATING

12. Will you like to share the knowledge of how do you do the work with the other personnel?

☐ Yes ☐ No

13. What knowledge is required in order to perform your work more effectively?

☐ Design for manufacturing constraints
☐ Machine capabilities
☐ Mould design and fabrication constraints
☐ Other: _____

14. What product life cycle data is required to support collaborative product development?

- ☐ Specification data
- ☐ Conceptual design
- ☐ 2D and 3D drawings
- ☐ Test data
- ☐ Tool design
- ☐ Bill of materials
- ☐ Work instructions
- ☐ Legacy documents
- ☐ Scanned documents
- ☐ Product assembly
- ☐ Parts and components
- ☐ Other_____

_____

15. Which software will support you to do your work more effectively?

- ☐ Design for manufacturing
- ☐ Selection of production equipment
- ☐ Mould design
- ☐ Mould fabrication
- ☐ Running CAD simultaneously
- ☐ Sessions to share a geometry
- ☐ Project Management
- ☐ Cost Modeling
- ☐ Product data repository
- ☐ Design history repository
- ☐ Testing services
- ☐ Other_____

_____

CULTURAL ISSUES FOR COLLABORATION

16. Is the culture of the companies situated in other countries a barrier to do the work in collaboration with them?

☐ Always ☐ Sometimes ☐ Never

17. Is the language a barrier to collaborate with the supply chain situated in other countries?

☐ Always ☐ Sometimes ☐ Never

18. Is the time a barrier to collaborate with the supply chain situated in other countries?

☐ Always ☐ Sometimes ☐ Never

Appendix B

IDEF0 Activity Modelling Technique

B.1 Introduction

This appendix presents an overview of the IDEF0 modelling technique. This technique has been used to model the activities developed during collaborative product development.

B.2 IDEF0 technique

IDEF0 was released in 1993 by the Computer Systems Laboratory of the National Institute of Standards and Technology (NIST 2004) as a standard for activity modelling. IDEF0 is a technique designed to model the activities of an organisation or system. As an analysis tool, IDEF0 assists the modeler in identifying what functions are performed, what is needed to perform those functions, what the current system does right, or wrong. It was derived from a graphical language, the Structured Analysis and Design Technique (SADT).

B.3 Graphical constructs of the IDEF0 technique

IDEF0 technique is a top-down hierarchical technique and it describes a system by a series of activities sequentially arranged. As shown in figure B.1, IDEF0 basic concepts include the following (Knowledge Based Systems Inc. 2000):

1. Graphic Representation

The "box and arrow" graphics of an IDEF0 diagram show an activity as a box and the interfaces to or from the activity as arrows entering or leaving the box (see figure B.1). The arrows represent the inputs, outputs, controls and mechanisms of those activities.

The inputs are transformed in the activity using, but not consuming, mechanisms or resources such as staff and machines to produce outputs. Typically the operation of the activity will be moderated by controls such as policies and procedures.

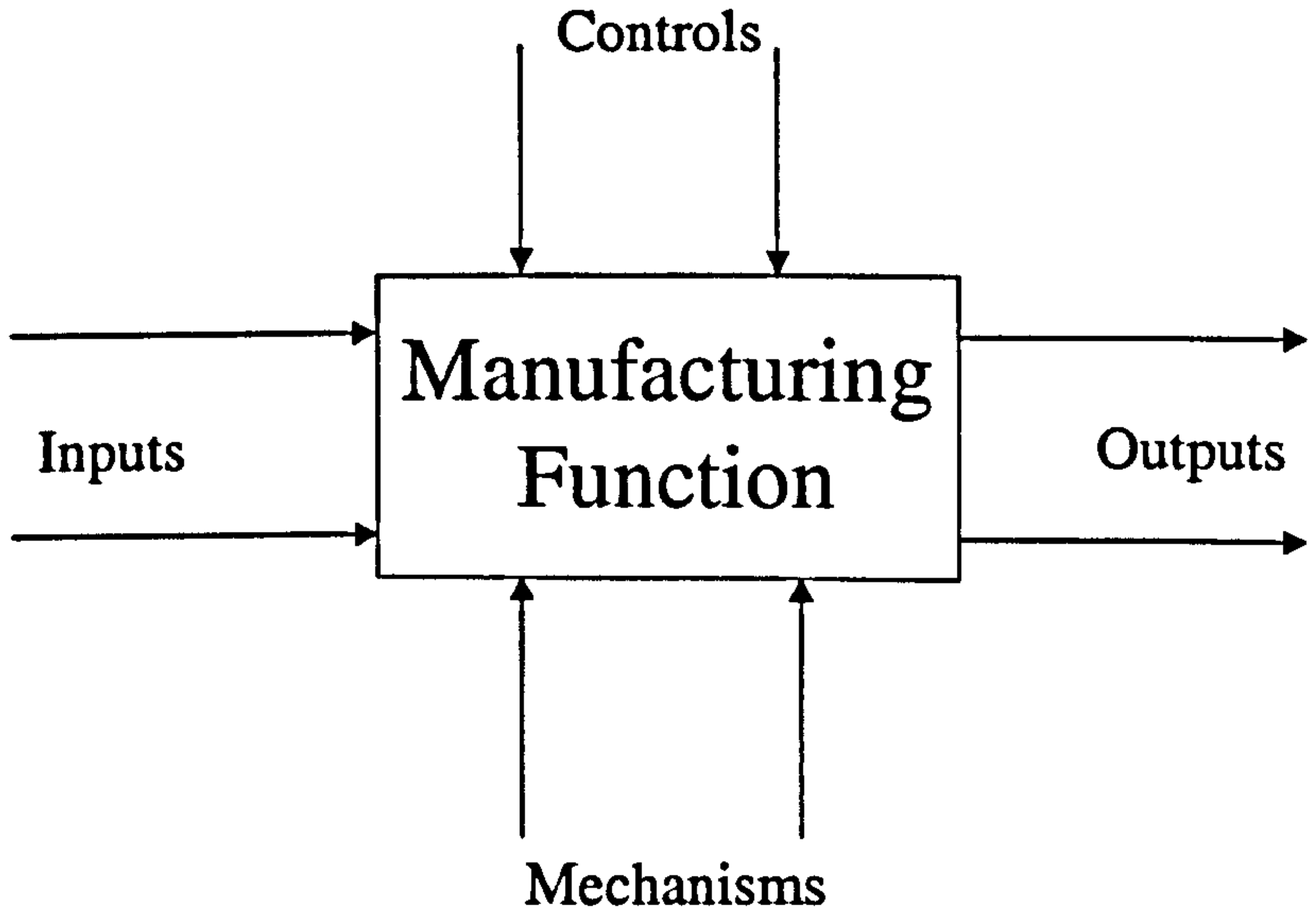


Figure B.1 IDEF0 function notation and interface arrows

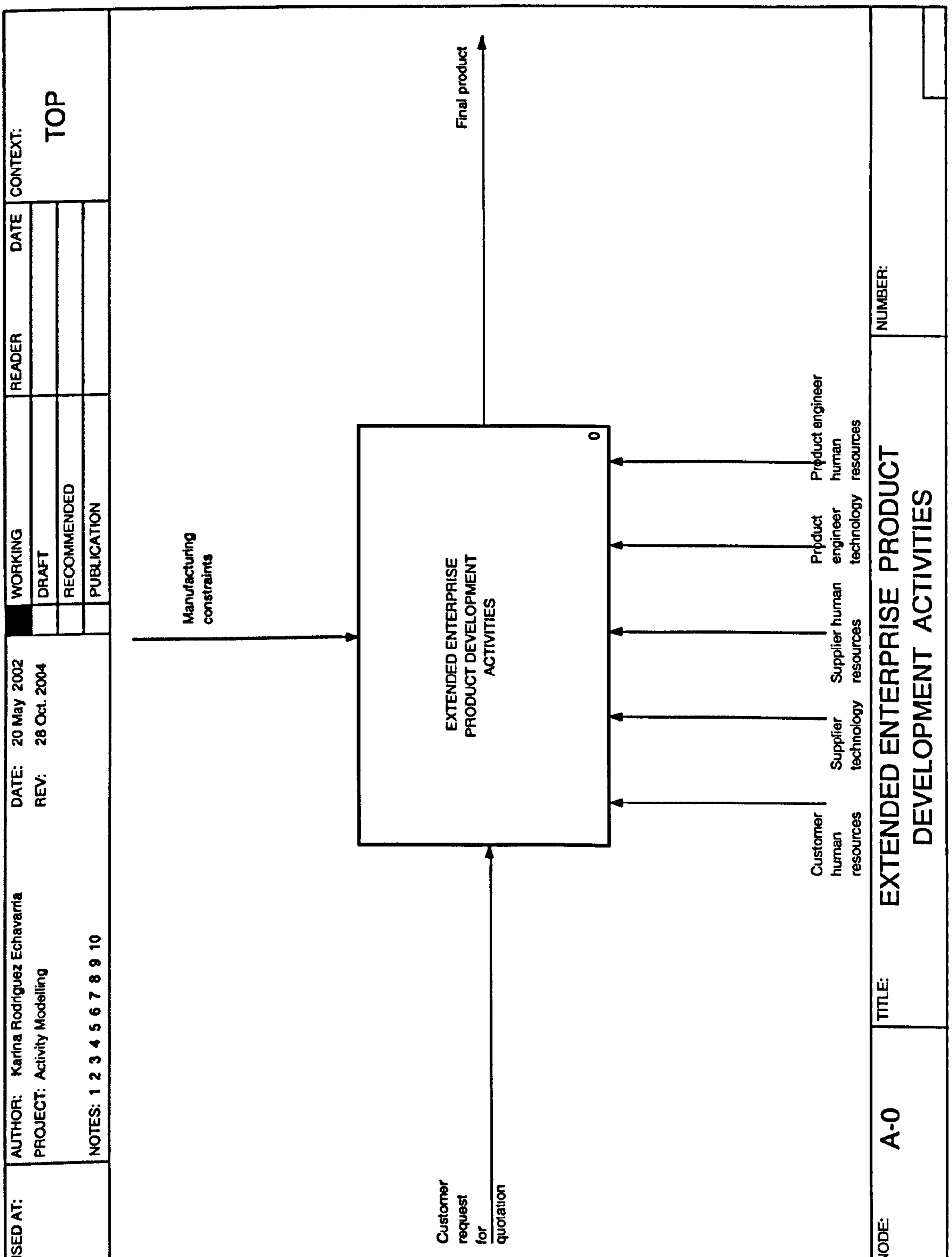
2. Communication

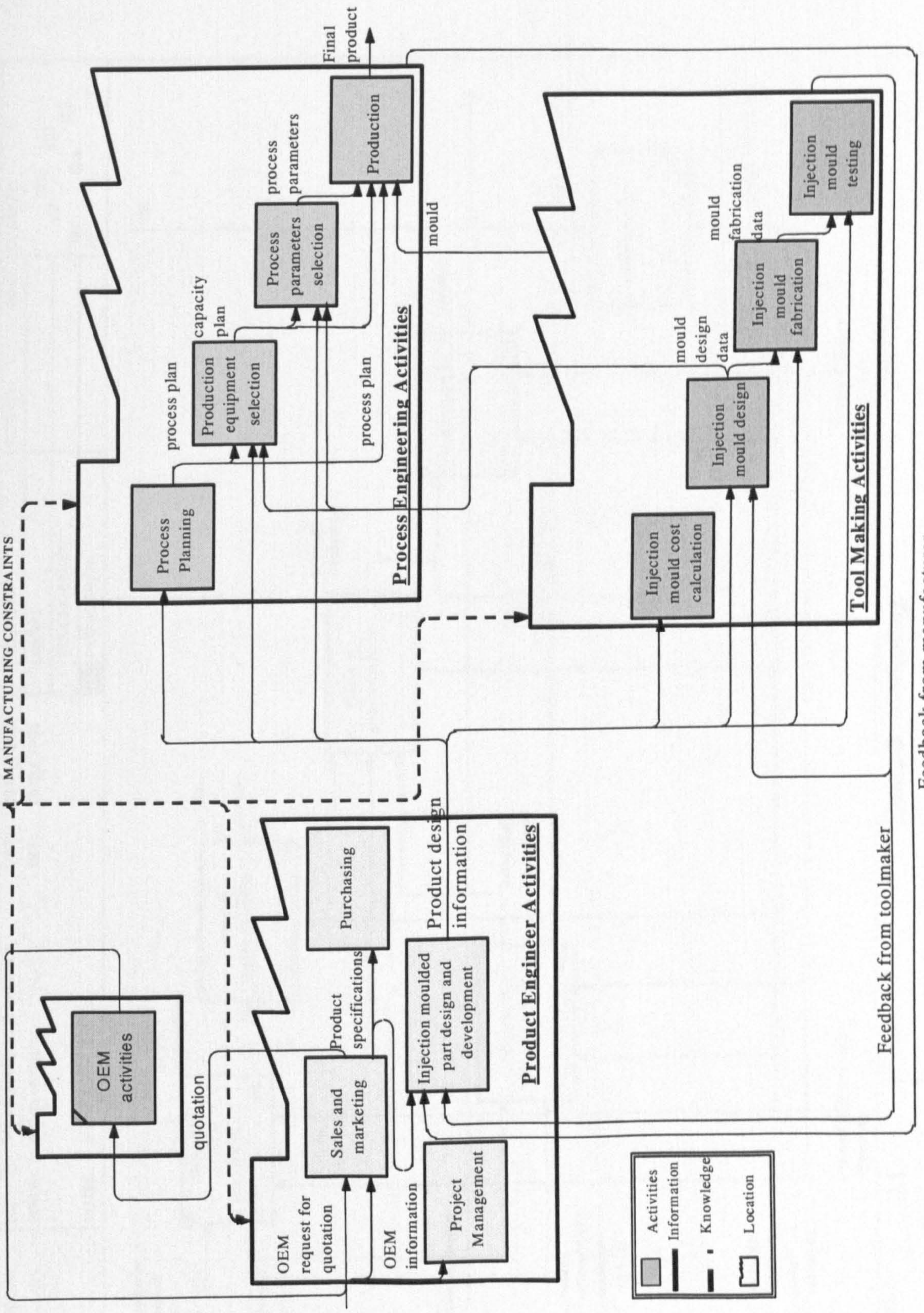
IDEF0 concepts designed to enhance communication include the following:

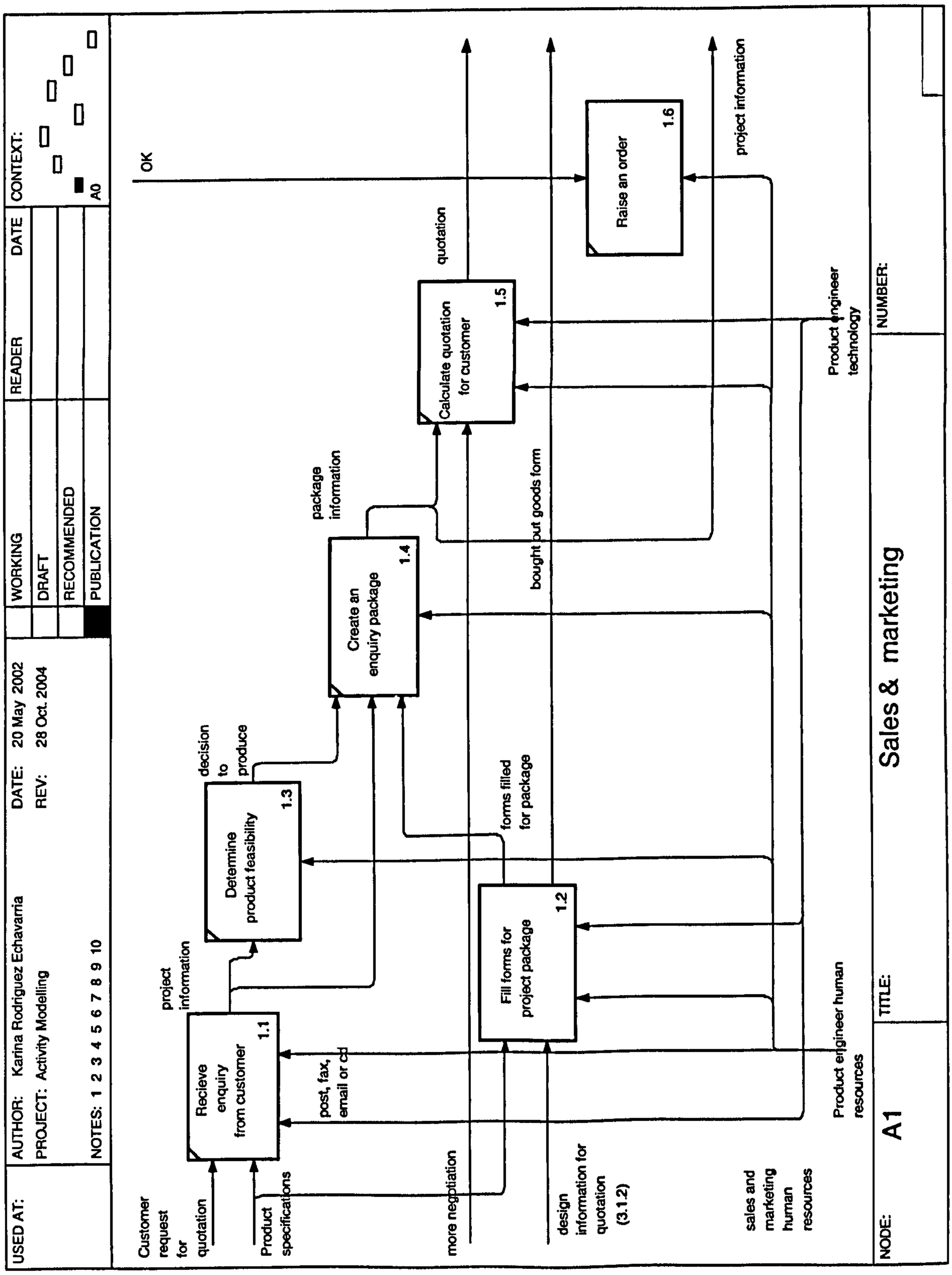
- Diagrams based on simple box and arrow graphics.
- English text labels to describe boxes and arrows and text to define the precise meanings of diagram elements.
- The gradual exposition of detail featuring a hierarchical structure, with the major activities at the top and with successive levels of sub activities revealing well-bounded detail breakout.

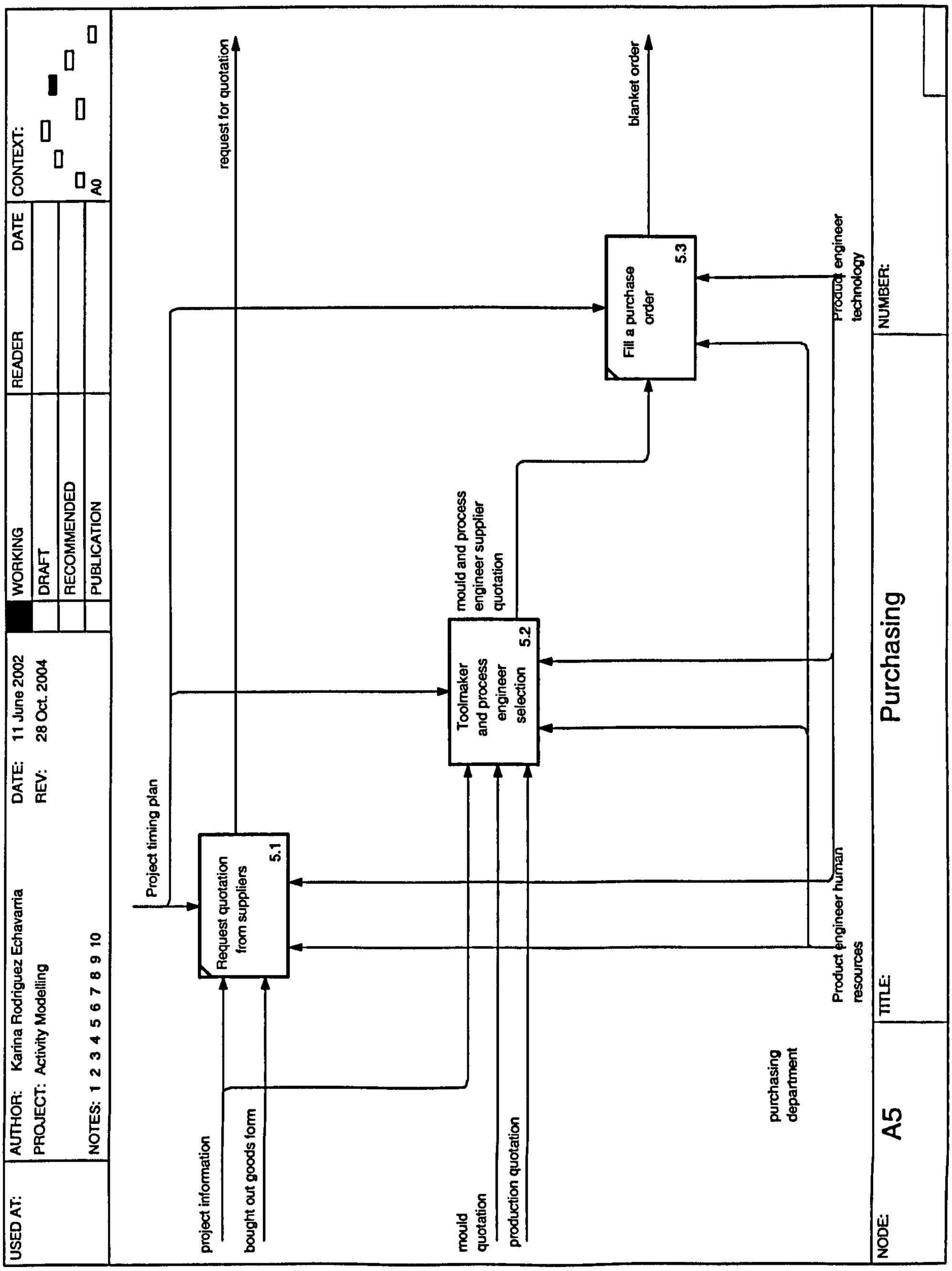
Appendix C

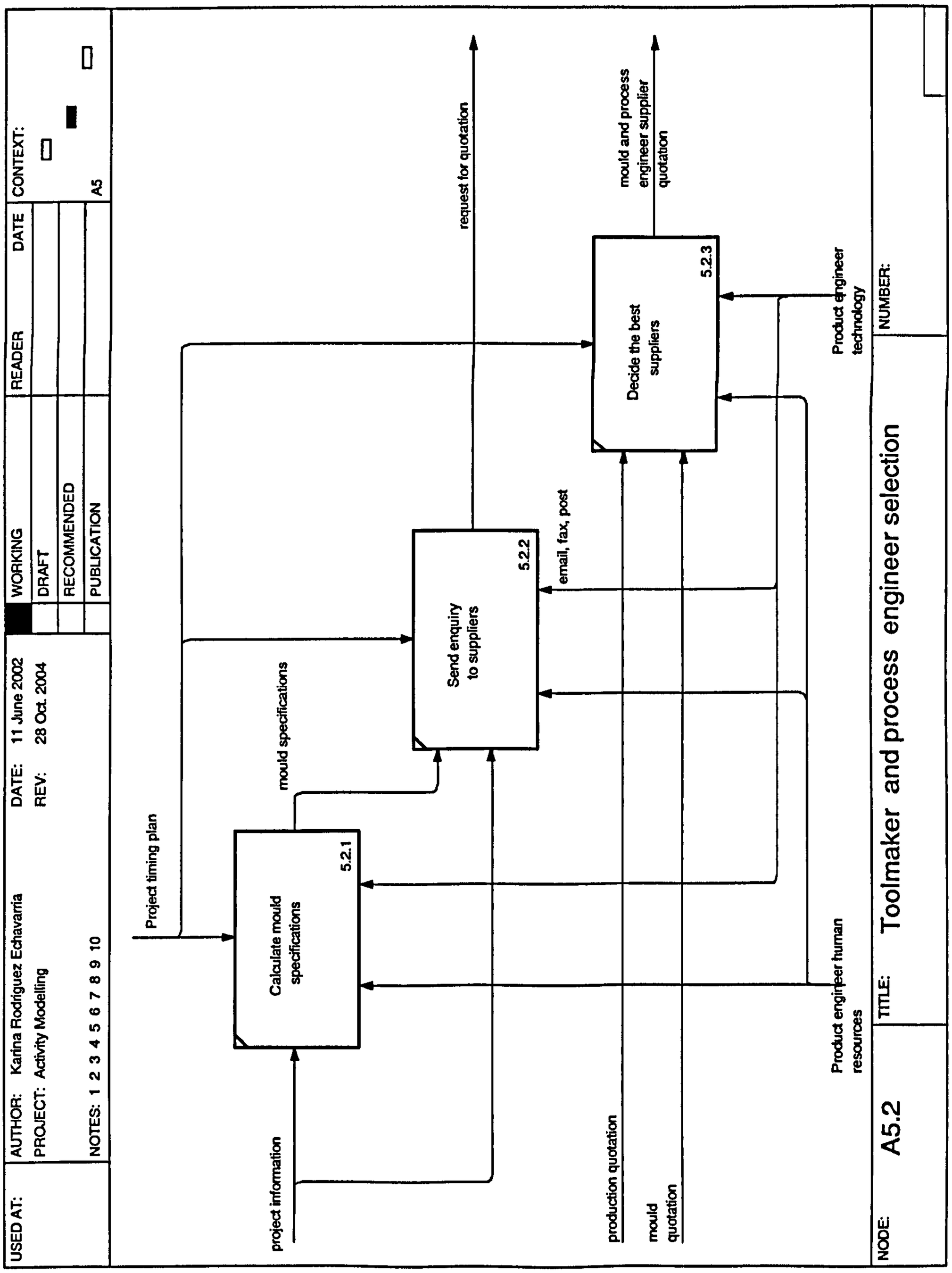
Extended Enterprise Product Development Activities

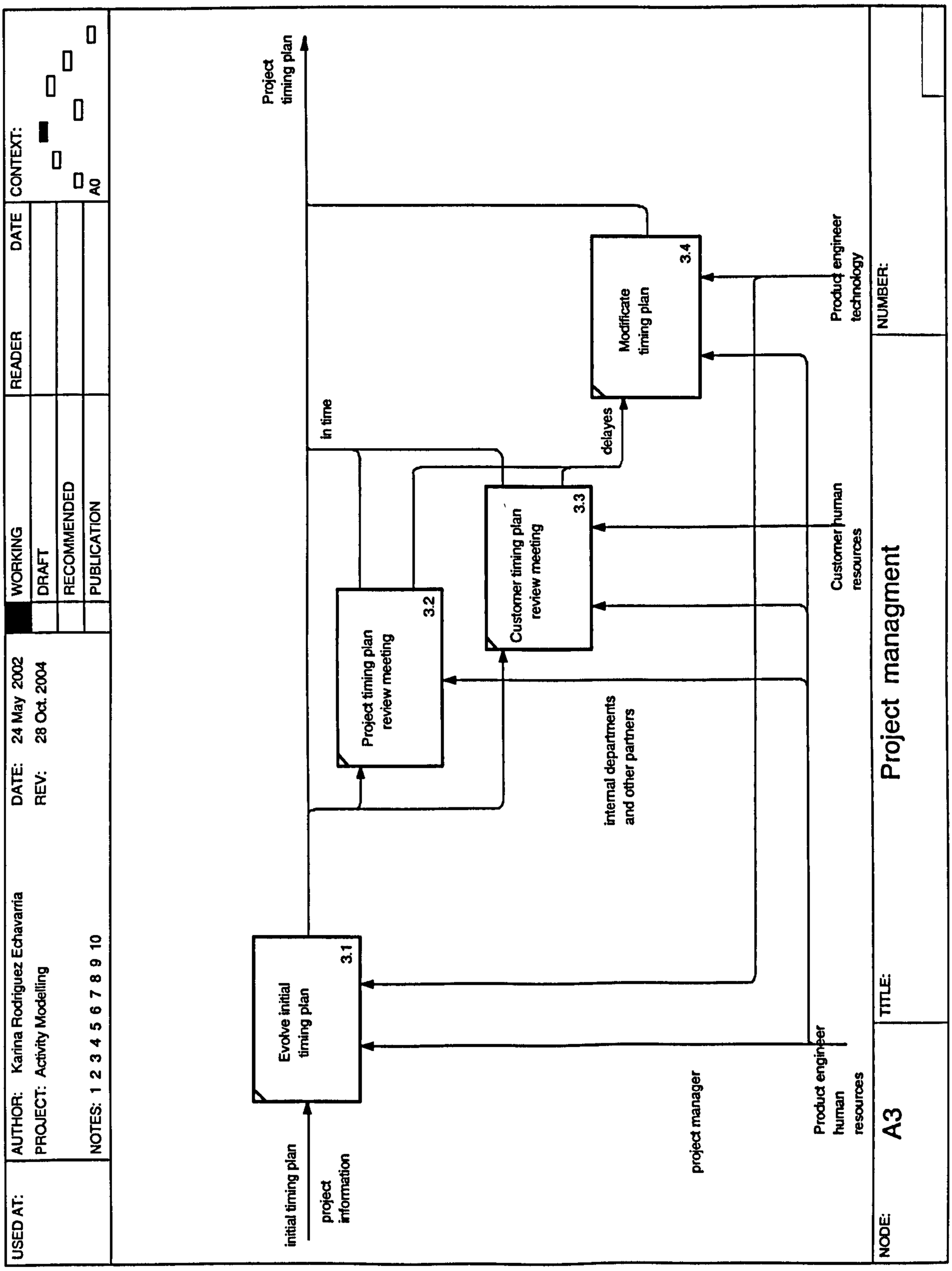

CONTEXT:
TOP
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
AUTHOR: Karina Rodriguez Echavarria
PROJECT: Activity Modelling
DATE: 20 May 2002
REV: 28 Oct. 2004
NOTES: 1 2 3 4 5 6 7 8 9 10
USED AT:
Final product
Manufacturing constraints
EXTENDED ENTERPRISE PRODUCT DEVELOPMENT ACTIVITIES
0
Product engineer human resources
Product engineer technology
Supplier human resources
Supplier technology
Customer human resources
Customer request for quotation
NUMBER:
TITLE: EXTENDED ENTERPRISE PRODUCT DEVELOPMENT ACTIVITIES
A-0
NODE:


MANUFACTURING CONSTRAINTS
Final product
Production
process parameters
Process parameters selection
capacity plan
Production equipment selection
process plan
Process Planning
Process Engineering Activities
mould
Injection mould testing
mould fabrication data
Injection mould fabrication
mould design data
Injection mould design
Injection mould cost calculation
Tool Making Activities
Feedback from manufacturer
OEM activities
quotation
Sales and marketing
Product specifications
Purchasing
Injection moulded part design and development
Product design information
Product Engineer Activities
Feedback from toolmaker
OEM request for quotation
OEM information
Project Management
Activities
Information
Knowledge
Location

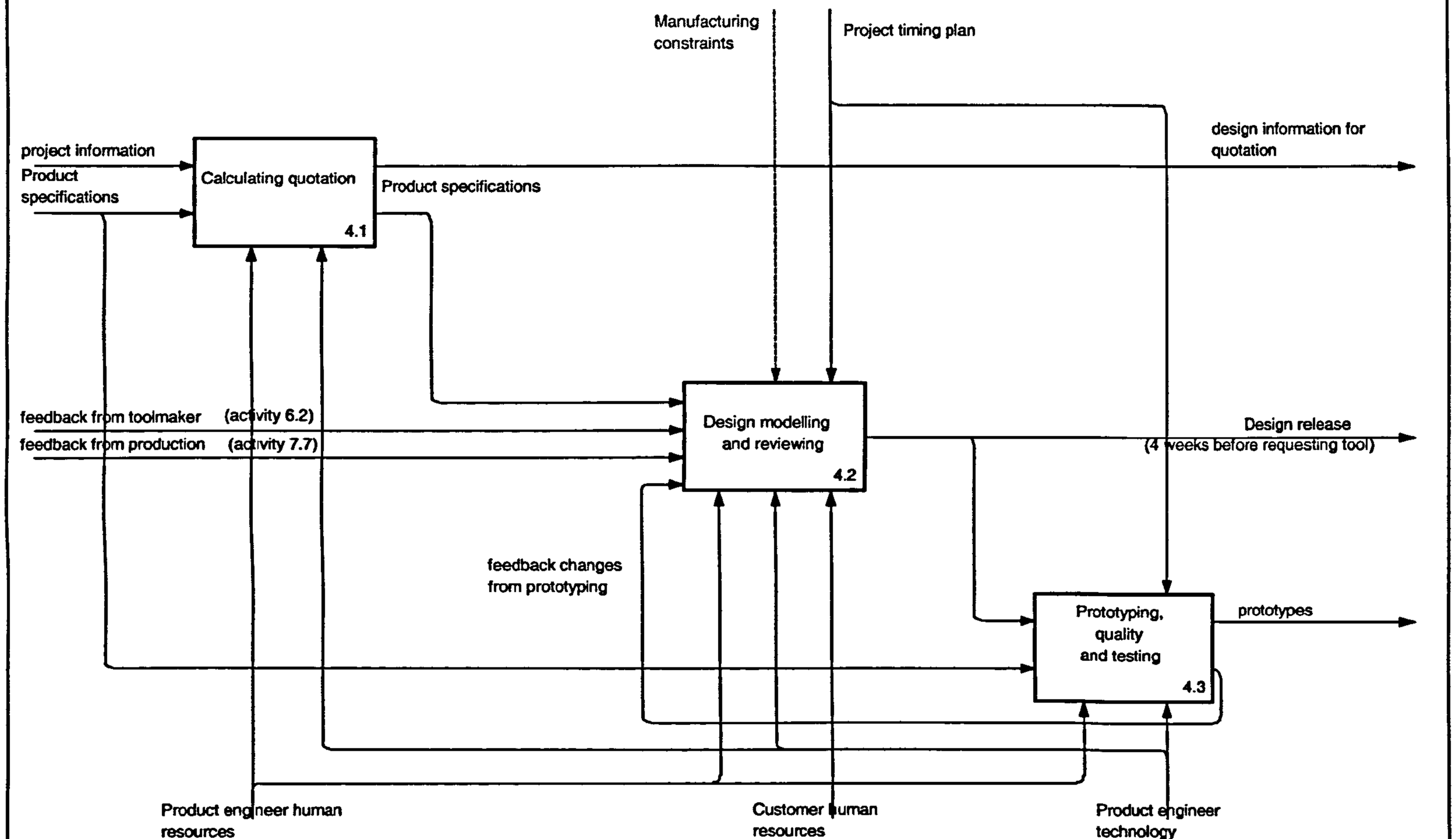

Sales & marketing
Product engineer technology
Product engineer human resources
Recieve enquiry from customer 1.1
Create an enquiry package 1.4
Determine product feasibility 1.3
Fill forms for project package 1.2
Calculate quotation for customer 1.5
Raise an order 1.6
project information
post, fax, email or cd
package information
decision to produce
forms filled for package
bought put goods form
quotation
OK
project information
Customer request for quotation
Product specifications
more negotiation
design information for quotation (3.1.2)
sales and marketing human resources
CONTEXT: A0
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
AUTHOR: Karina Rodriguez Echavarria
PROJECT: Activity Modelling
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 20 May 2002
REV: 28 Oct. 2004
USED AT:
A1
TITLE:
NUMBER:
NODE:


CONTEXT:
A0
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
AUTHOR: Karina Rodriguez Echavarria
PROJECT: Activity Modelling
DATE: 11 June 2002
REV: 28 Oct. 2004
NOTES: 1 2 3 4 5 6 7 8 9 10
request for quotation
blanket order
Fill a purchase order 5.3
mould and process engineer supplier quotation
5.2
Product engineer technology
NUMBER:
Purchasing
Project timing plan
Request quotation from suppliers 5.1
Toolmaker and process engineer selection 5.2
Product engineer human resources
purchasing department
A5
TITLE:
project information
bought out goods form
mould quotation
production quotation
USED AT:
NODE:

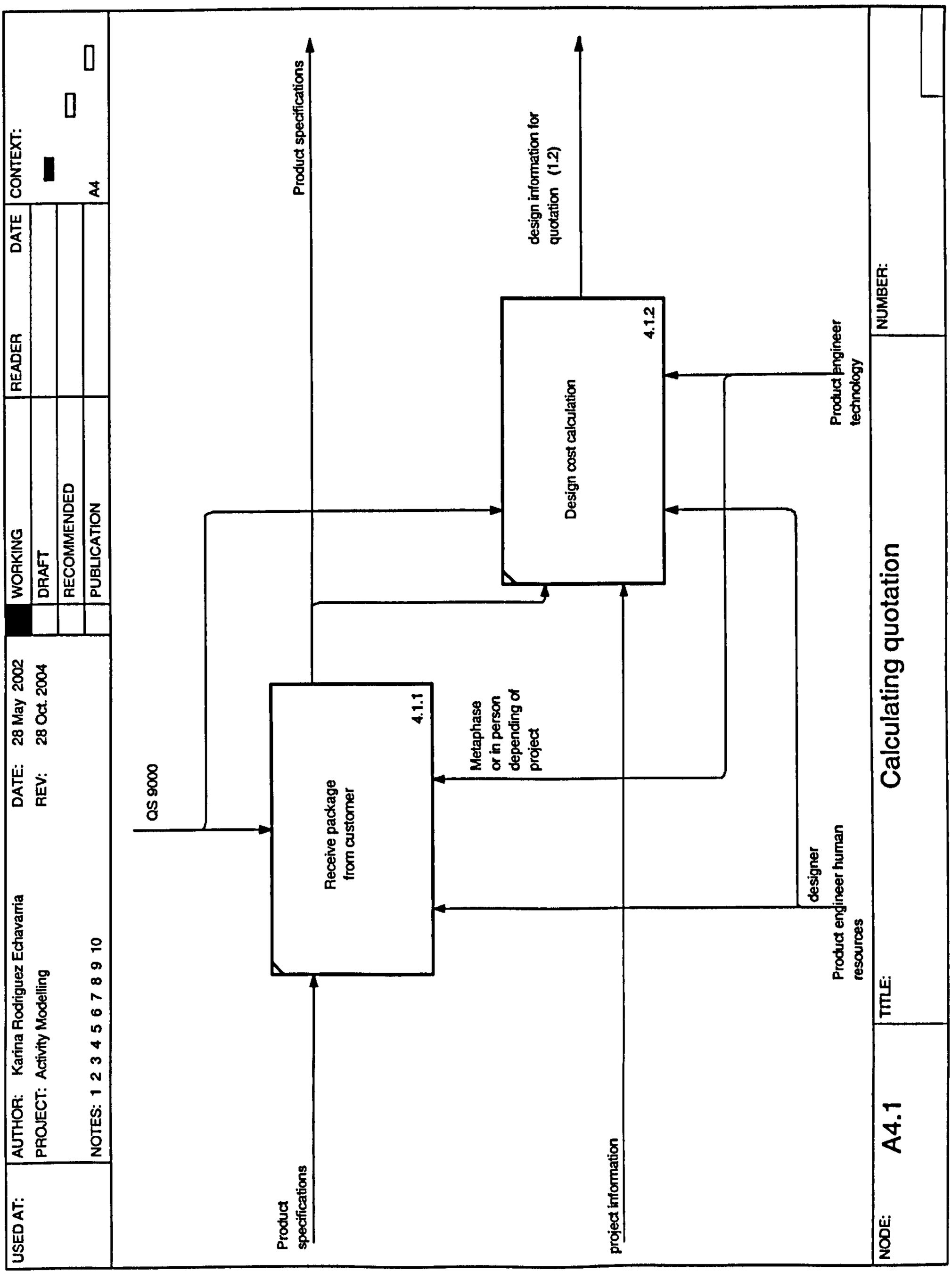

WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A5
AUTHOR: Karina Rodriguez Echavarria
PROJECT: Activity Modelling
DATE: 11 June 2002
REV: 28 Oct. 2004
NOTES: 1 2 3 4 5 6 7 8 9 10
USED AT:
request for quotation
mould and process engineer supplier quotation
Decide the best suppliers
5.2.3
Product engineer technology
NUMBER:
email, fax, post
Send enquiry to suppliers
5.2.2
Project timing plan
mould specifications
Calculate mould specifications
5.2.1
Product engineer human resources
TITLE: Toolmaker and process engineer selection
project information
production quotation
mould quotation
NODE: A5.2


CONTEXT:
A0
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
AUTHOR: Karina Rodriguez Echavarria
PROJECT: Activity Modelling
DATE: 24 May 2002
REV: 28 Oct. 2004
NOTES: 1 2 3 4 5 6 7 8 9 10
USED AT:
Project timing plan
in time
delayes
Modificate timing plan
3.4
Customer timing plan review meeting
3.3
Project timing plan review meeting
3.2
Evolve initial timing plan
3.1
initial timing plan
project information
Product engineer technology
Customer human resources
internal departments and other partners
project manager
Product engineer human resources
NUMBER:
TITLE:
Project managment
NODE:
A3

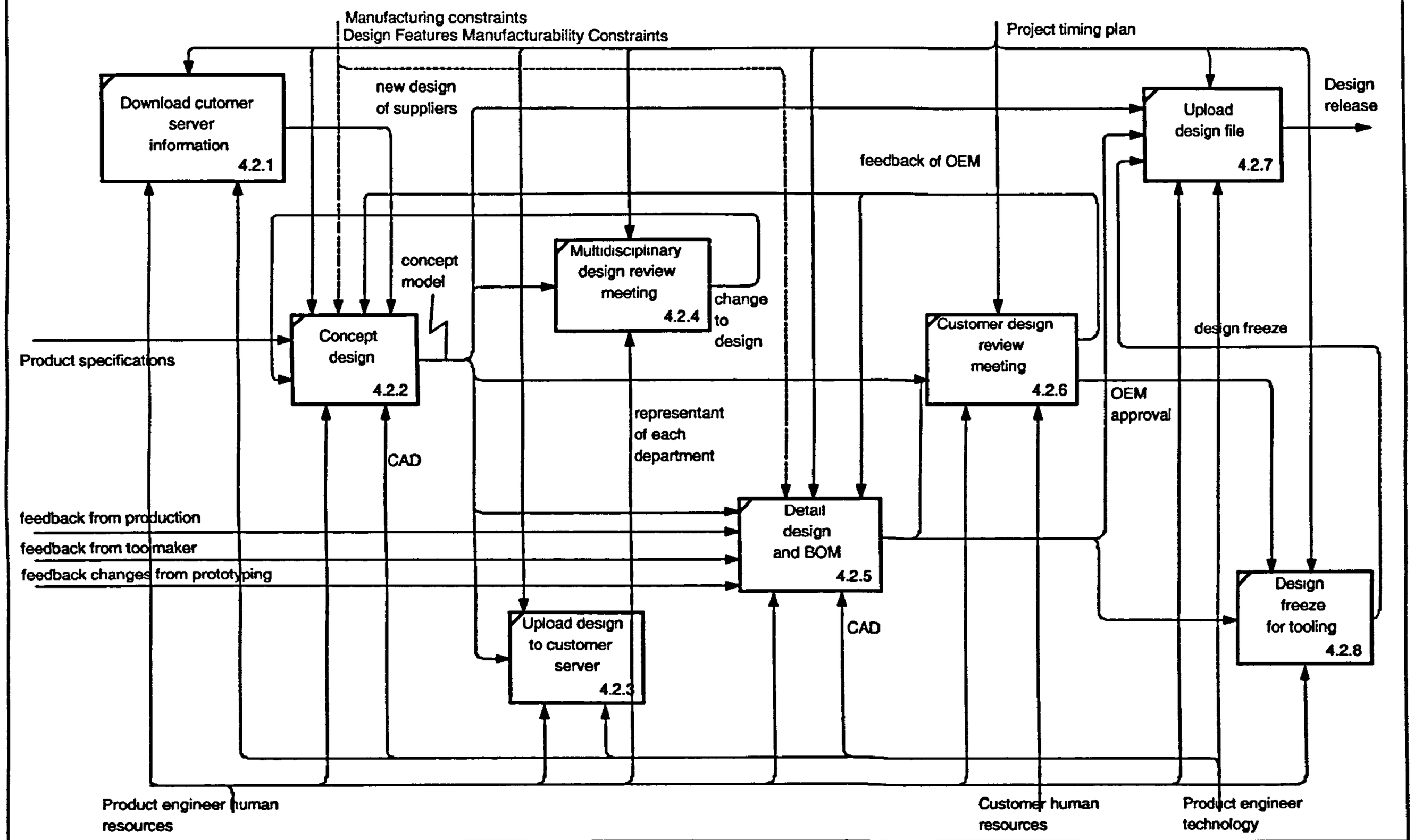
USED AT:	AUTHOR: Karina Rodriguez Echavarria	DATE: 20 May 2002	■ WORKING	READER DATE	CONTEXT:
	PROJECT: Activity Modelling	REV: 28 Oct. 2004	DRAFT		
			RECOMMENDED		
	NOTES: 1 2 3 4 5 6 7 8 9 10		PUBLICATION		A0



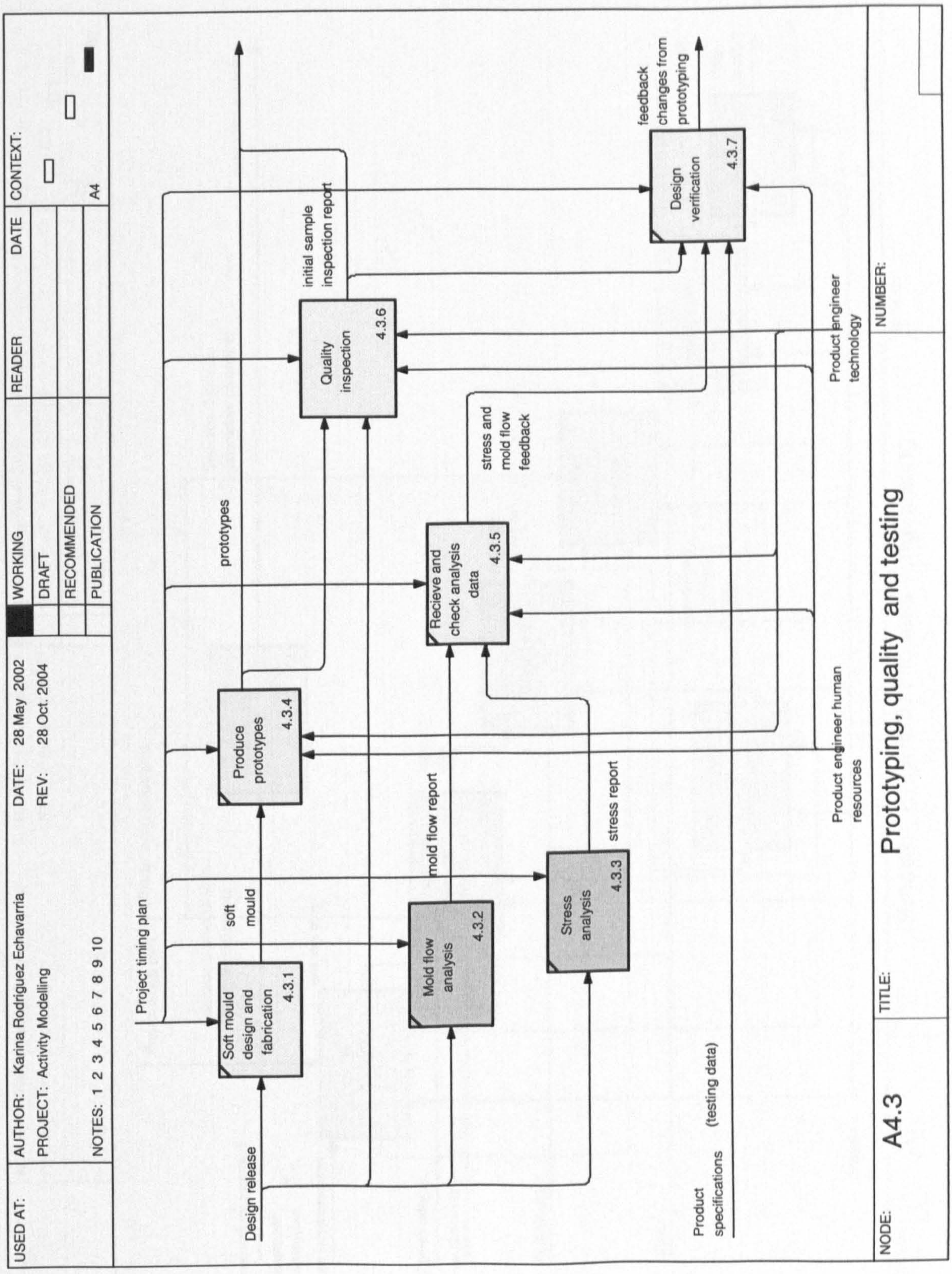
NODE: A4	TITLE: Injection moulded part design & development	NUMBER:


WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A4
AUTHOR: Karina Rodriguez Echavarria
PROJECT: Activity Modelling
DATE: 28 May 2002
REV: 28 Oct. 2004
NOTES: 1 2 3 4 5 6 7 8 9 10
USED AT:
Product specifications
project information
Product specifications
design information for quotation (1.2)
QS 9000
Metaphase or in person depending of project
Product engineer technology
designer
Product engineer human resources
Receive package from customer
4.1.1
Design cost calculation
4.1.2
NUMBER:
TITLE:
Calculating quotation
NODE:
A4.1

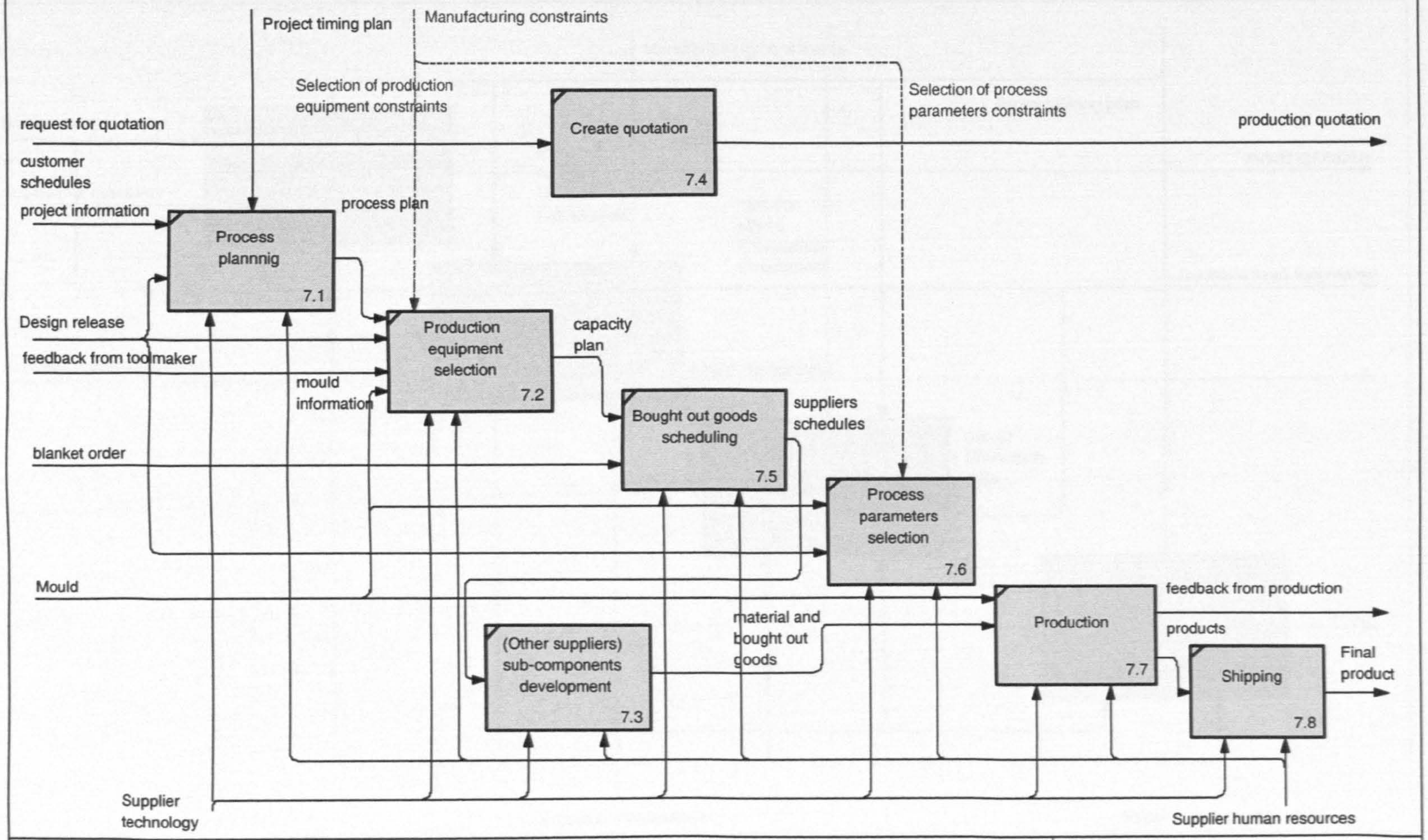
USED AT:	AUTHOR: Karina Rodriguez Echavarria	DATE: 28 May 2002	■ WORKING	READER DATE	CONTEXT:
	PROJECT: Activity Modelling	REV: 28 Oct. 2004	DRAFT		
			RECOMMENDED		
	NOTES: 1 2 3 4 5 6 7 8 9 10		PUBLICATION		A4



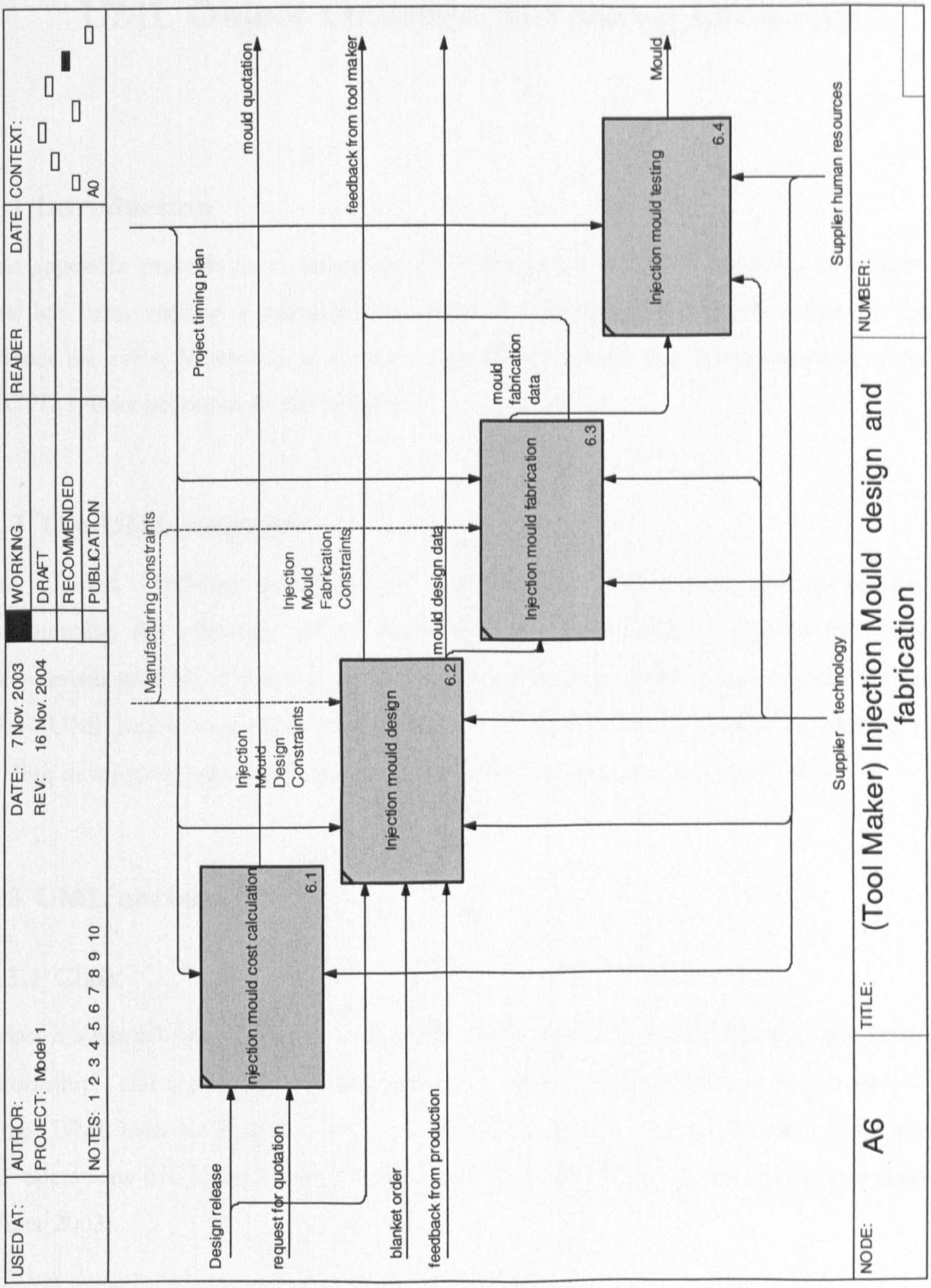
NODE: A4.2	TITLE: Design modelling and reviewing	NUMBER:


A-0
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A4
AUTHOR: Karina Rodriguez Echavarria
PROJECT: Activity Modelling
DATE: 28 May 2002
REV: 28 Oct. 2004
NOTES: 1 2 3 4 5 6 7 8 9 10
USED AT:
Project timing plan
prototypes
initial sample inspection report
feedback changes from prototyping
stress and mold flow feedback
soft mould
mold flow report
stress report
Design release
Soft mould design and fabrication 4.3.1
Produce prototypes 4.3.4
Quality inspection 4.3.6
Design verification 4.3.7
Recieve and check analysis data 4.3.5
Mold flow analysis 4.3.2
Stress analysis 4.3.3
Product engineer technology
Product engineer human resources
Product specifications (testing data)
NUMBER:
TITLE: Prototyping, quality and testing
NODE: A4.3

USED AT:	AUTHOR: Karina Rodriguez Echavarria	DATE: 11 June 2002		WORKING	READER DATE	CONTEXT:
	PROJECT: Activity Modelling	REV: 28 Oct. 2004		DRAFT		
				RECOMMENDED		
	NOTES: 1 2 3 4 5 6 7 8 9 10			PUBLICATION		A0



NODE: A7	TITLE: (Process Engineer) Process Engineering	NUMBER:


A-0
Injection Mould design and fabrication
Project timing plan
mould quotation
feedback from tool maker
Mould
Supplier human resources
Supplier technology
Manufacturing constraints
Injection Mould Fabrication Constraints
mould design data
mould fabrication data
Injection Mould Design Constraints
Design release
request for quotation
blanket order
feedback from production
injection mould cost calculation
6.1
Injection mould design
6.2
Injection mould fabrication
6.3
Injection mould testing
6.4
(Tool Maker) Injection Mould design and fabrication
A6
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A0
AUTHOR:
PROJECT: Model 1
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 7 Nov. 2003
REV: 16 Nov. 2004
USED AT:
TITLE:
NUMBER:
NODE:

Appendix D

UML Object Oriented Modelling Language

D.1 Introduction

This appendix presents an overview of the UML object oriented modeling technique. This has been used as a technique to model the knowledge and information of the product life cycle. Moreover, it has been used for the design and implementation of the KdCPD system proposed in this research.

D.2 The UML language

The Unified Modeling Language is a technique for specifying, visualising, and documenting the elements of an object-oriented system under development. One characteristic of UML is that it is methodology-independent (Object Management Group 2004). UML builds on previous notational methods such as Booch, OMT, and OOSE. It is being developed under the auspices of the Object Management Group (OMG).

D.3 UML notation

D.3.1 Class

A class is a named description of a set of objects that share the same attributes, operations, relationships, and semantics. These objects can represent real-world things or conceptual things. UML class notation is a rectangle divided into three parts: class name, attributes, and operations (see figure D1). Relationships between classes are the connecting links (Miller 2003).

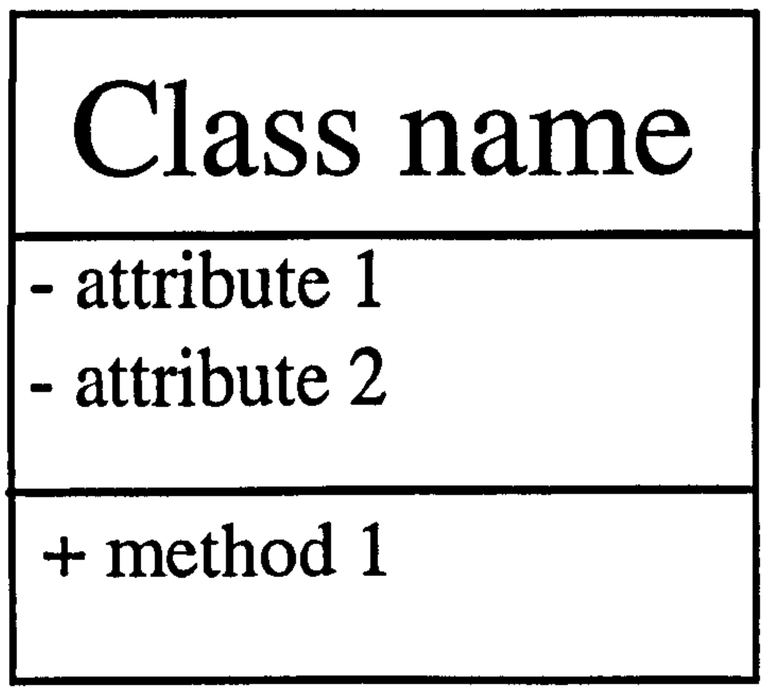


Figure D.1 UML notation of a class with attributes and methods

D.3.2 Class relationships

Class diagrams has mainly three kinds of relationships (Miller 2003):

1. Aggregation: an association in which one class belongs to a collection. An aggregation has a diamond end pointing to the part containing the whole, as shown in figure D.2

2. Generalisation: an inheritance link indicating one class is a superclass of the other. Figure D.2 shows how a generalisation has a triangle pointing to the superclass.

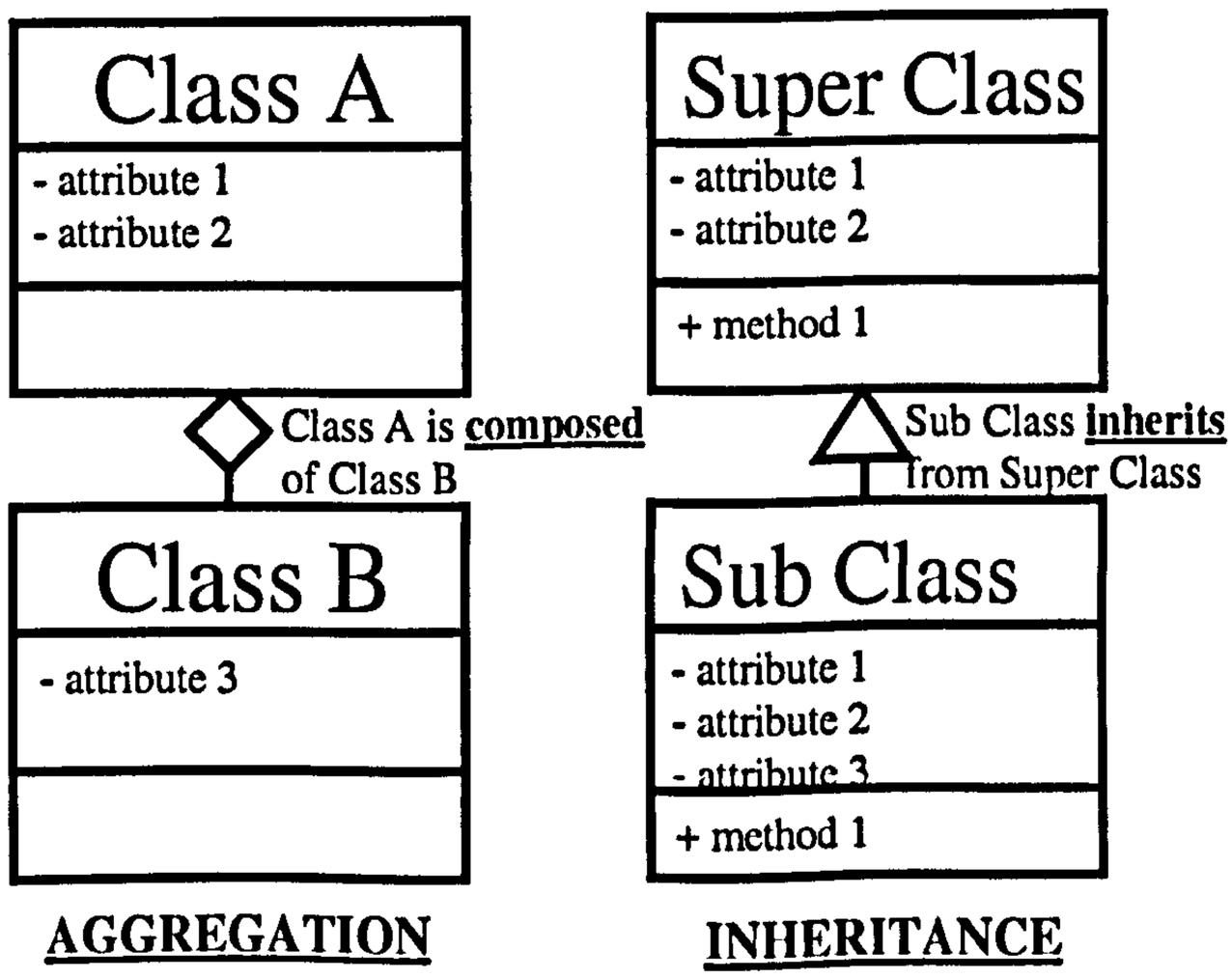
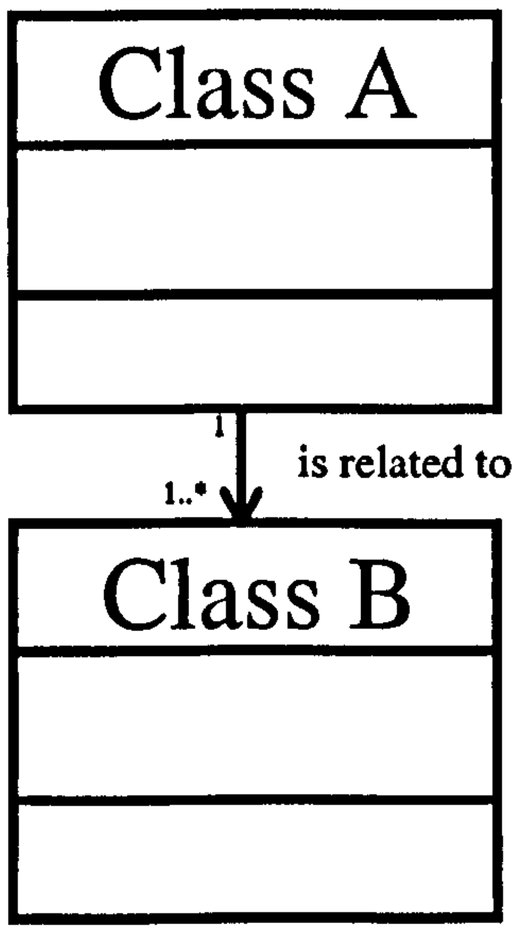


Figure D.2 UML notation of aggregation and generalisation relationships

3. Association: a relationship between instances of the two classes as illustrated in figure D.3 There is an association between two classes if an instance of one class must know about the other in order to perform its work. In a diagram, an association is a link connecting two classes.



ASSOCIATION

Figure D.3 UML notation of an association relationship

A navigability arrow on an association shows which direction the association can be traversed or queried. The arrow also indicates who "owns" the association's implementation. Associations with no navigability arrows are bi-directional.

The multiplicity of an association end is the number of possible instances of the class associated with a single instance of the other end. Multiplicities are single numbers or ranges of numbers. Table D.1 gives the most common multiplicities.

Table D.1 UML notation for association's multiplicities

Multiplicities	**Meaning**
0..1	Zero or one instance. The notation ***n*** **. .** ***m*** indicates ***n*** to ***m*** instances.
0..* *or* *****	No limit on the number of instances (including none).
1	Exactly one instance
1..*	At least one instance

D.3.3 Package

To simplify complex class diagrams, classes could be grouped into packages. As shown in figure D.4 a package is a collection of logically related UML elements. Packages appear as rectangles with small tabs at the top. The package name is on the tab or inside the rectangle (Miller 2003).

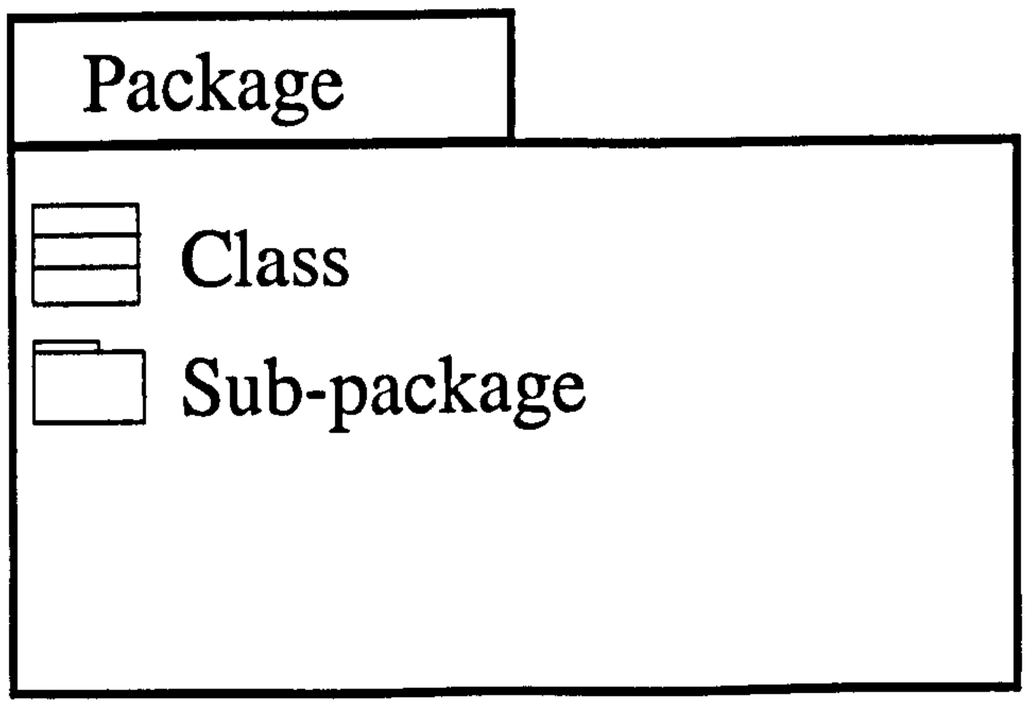


Figure D.4 UML notation of a package

D.3.4 Object

An object is an instance of a class. It is represented as a rectangle, which corresponds to a single instance. As shown in figure D.5 instance names are underlined in UML diagrams (Miller 2003).

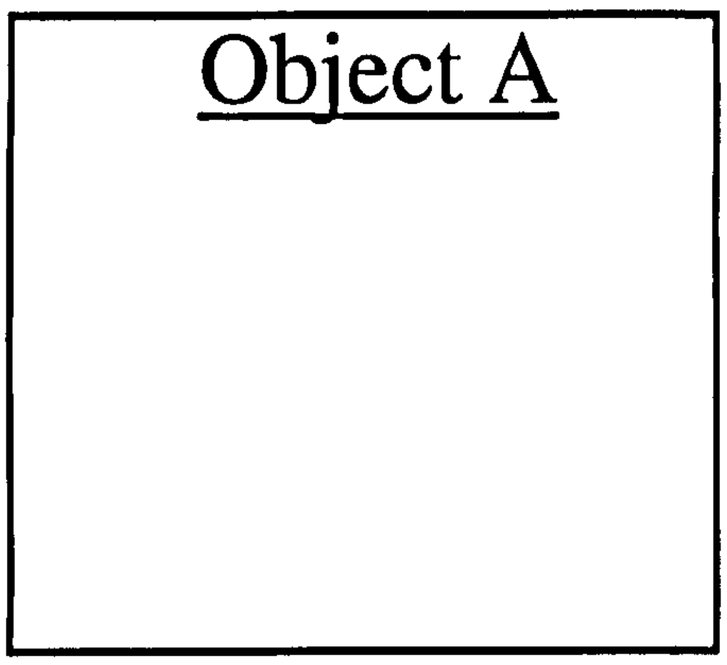
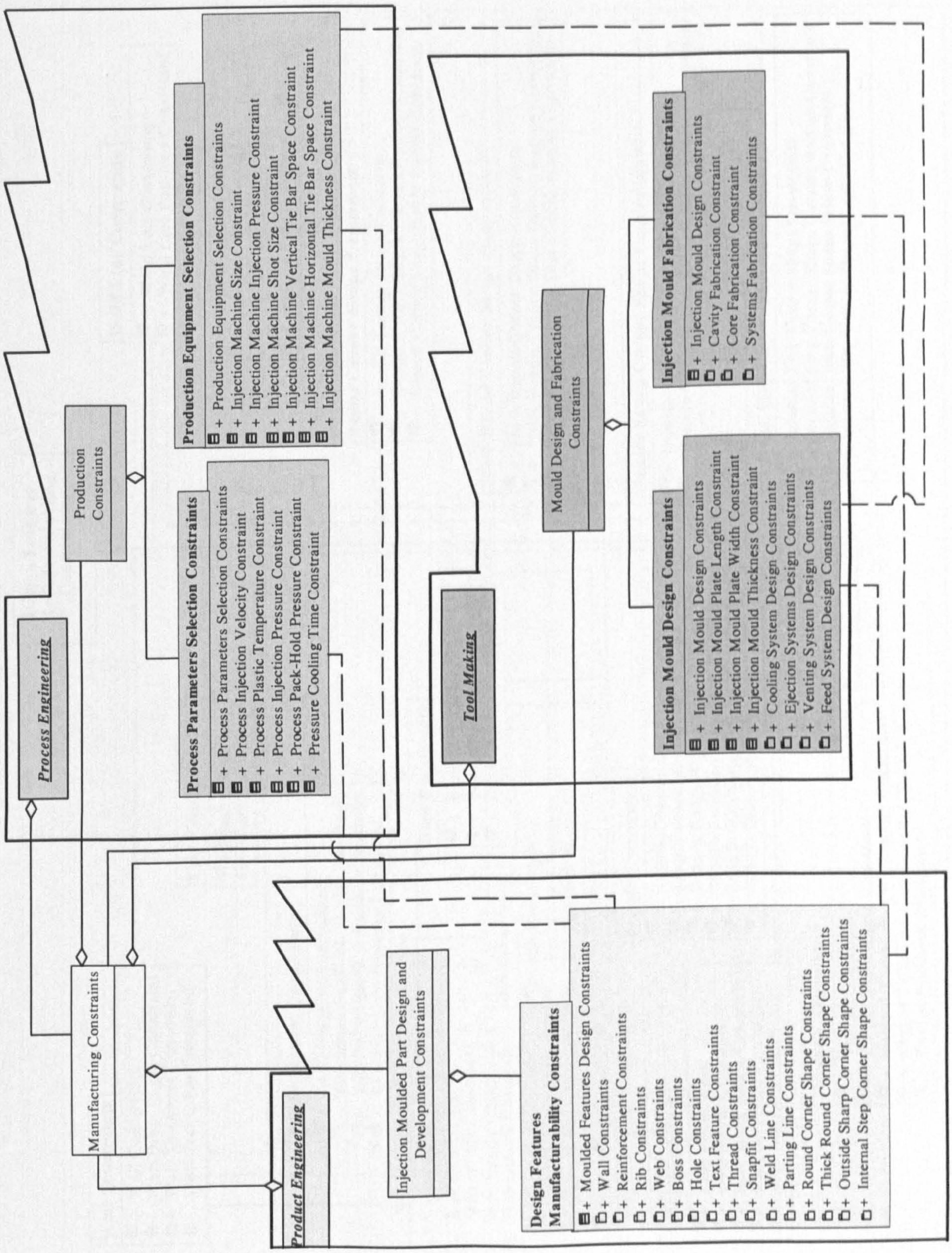


Figure D.5 UML notation of an object

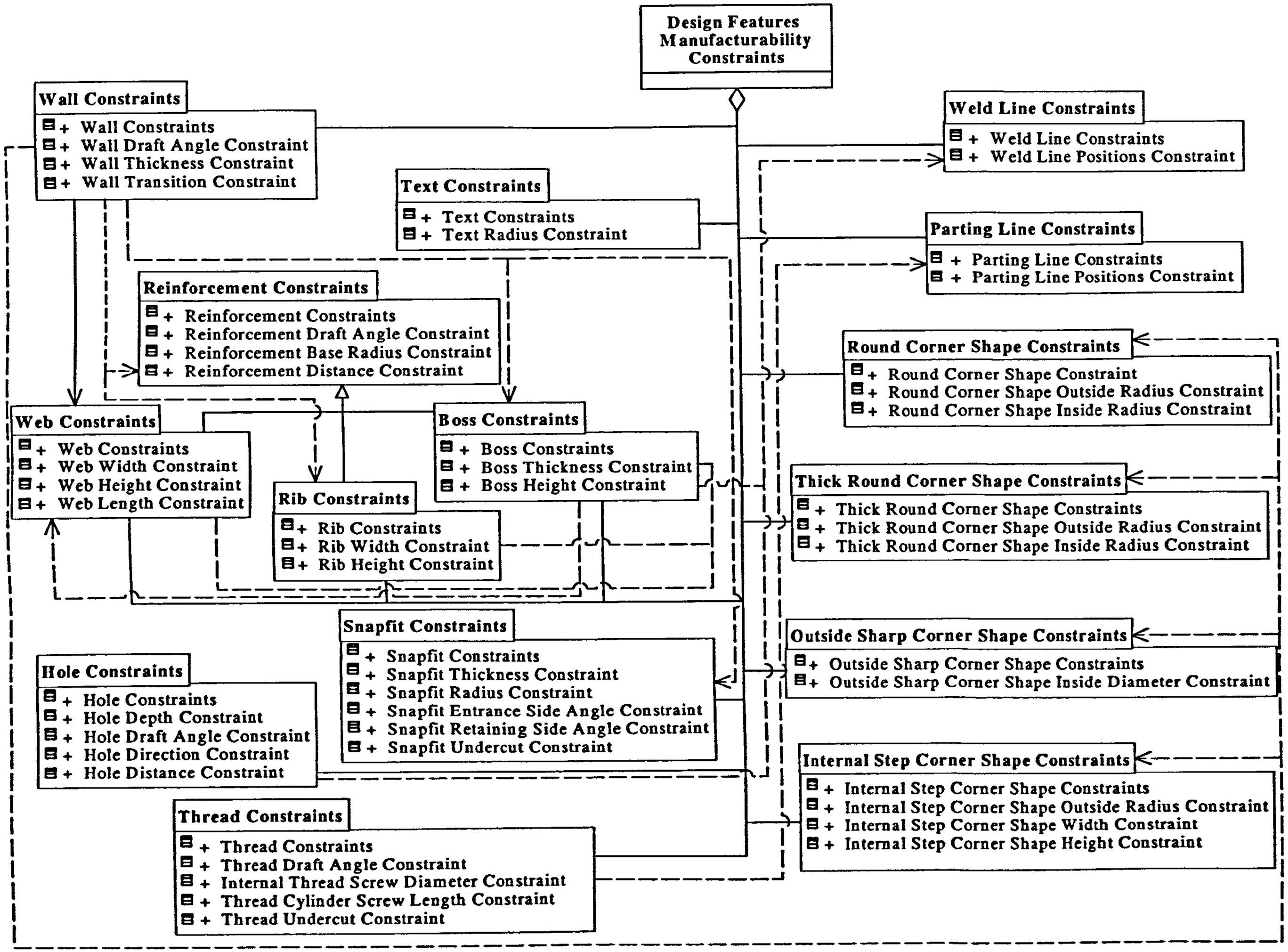
Appendix E

Manufacturing Knowledge Model

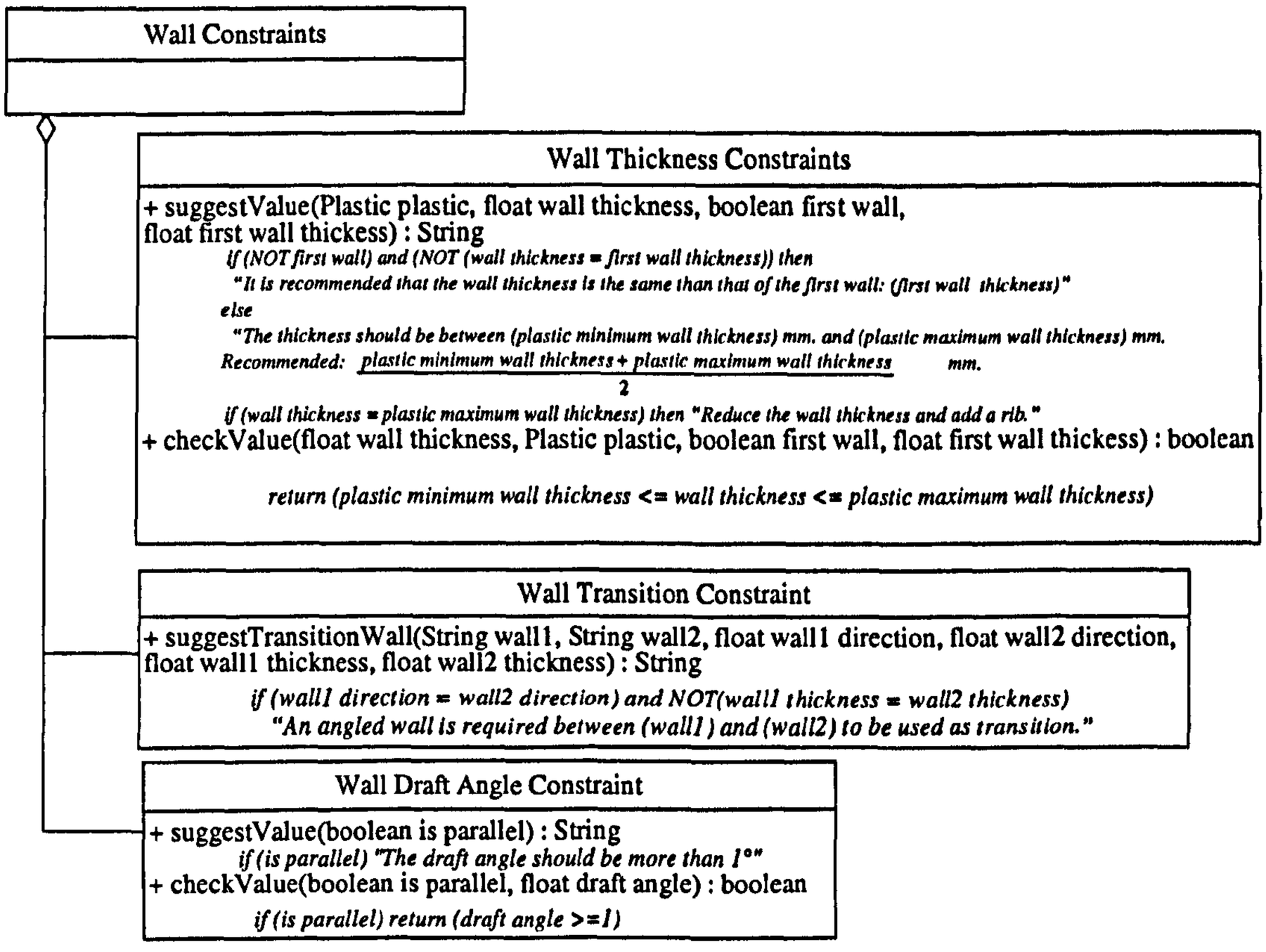
E.1 Manufacturing Knowledge Model top view



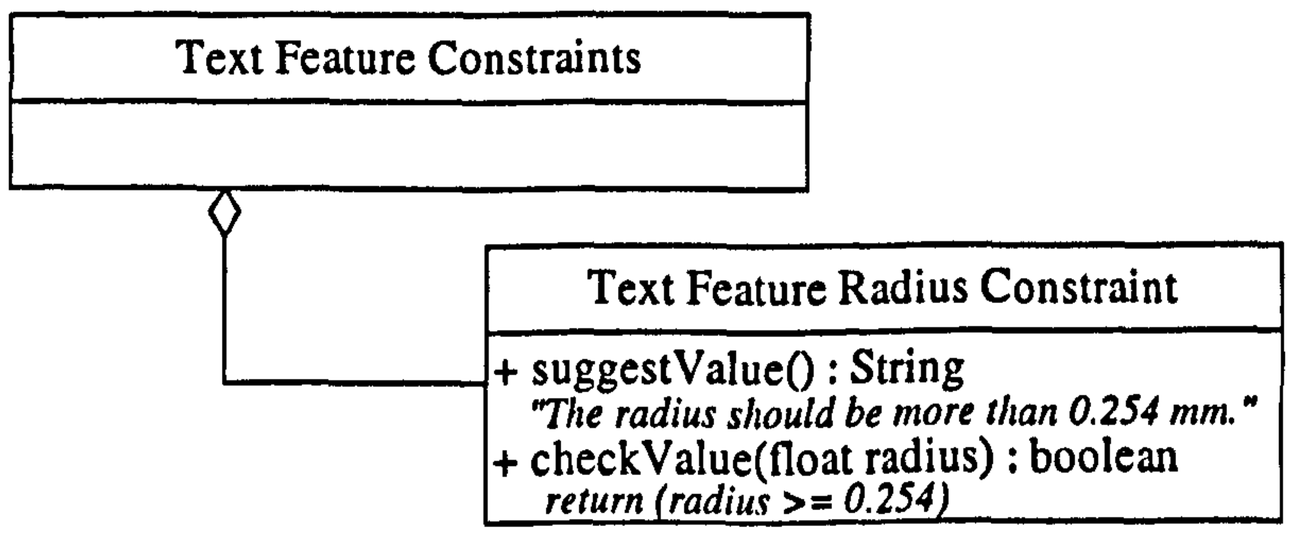
E.2 Design features manufacturability constraints



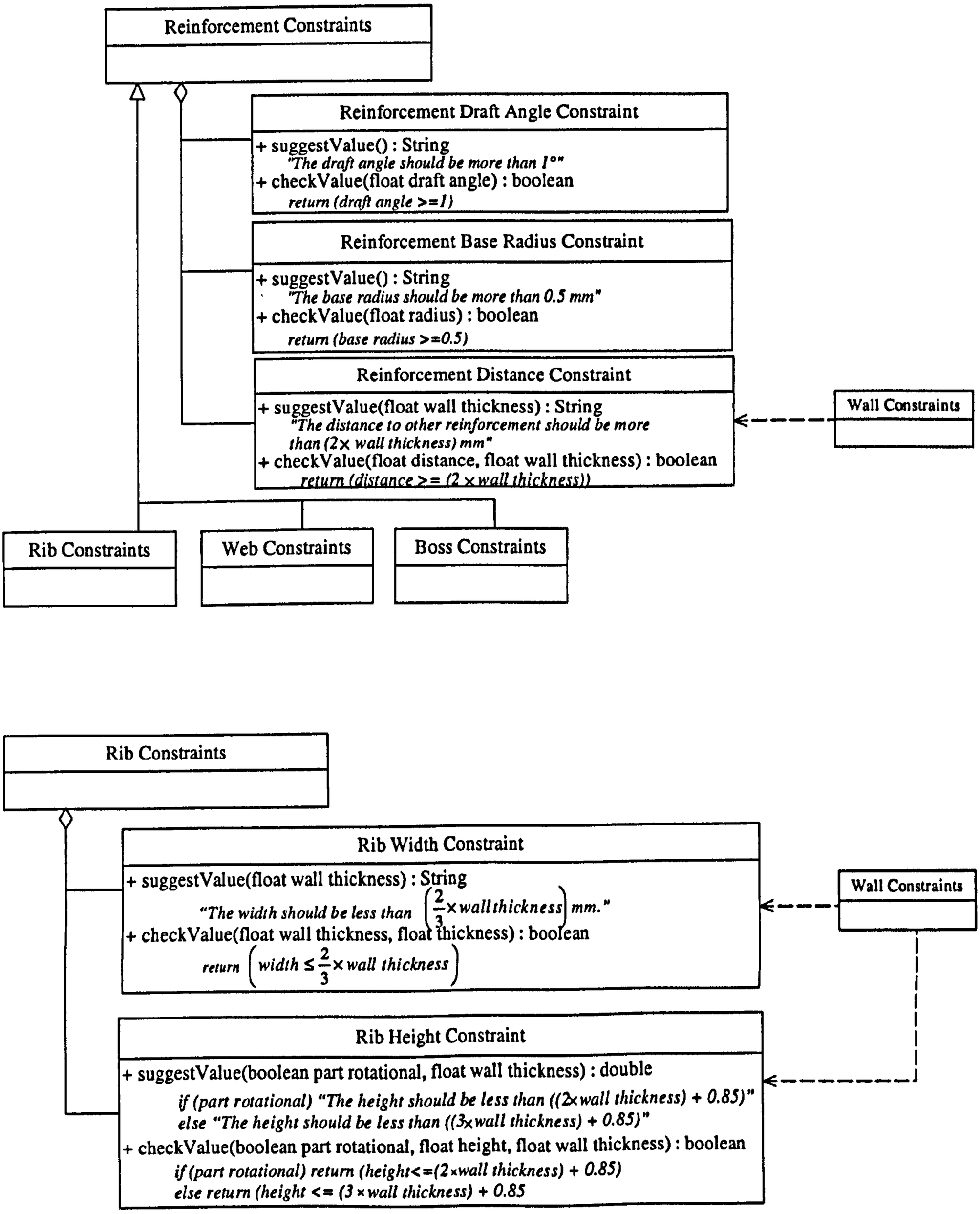
E.2.1 Wall constraints

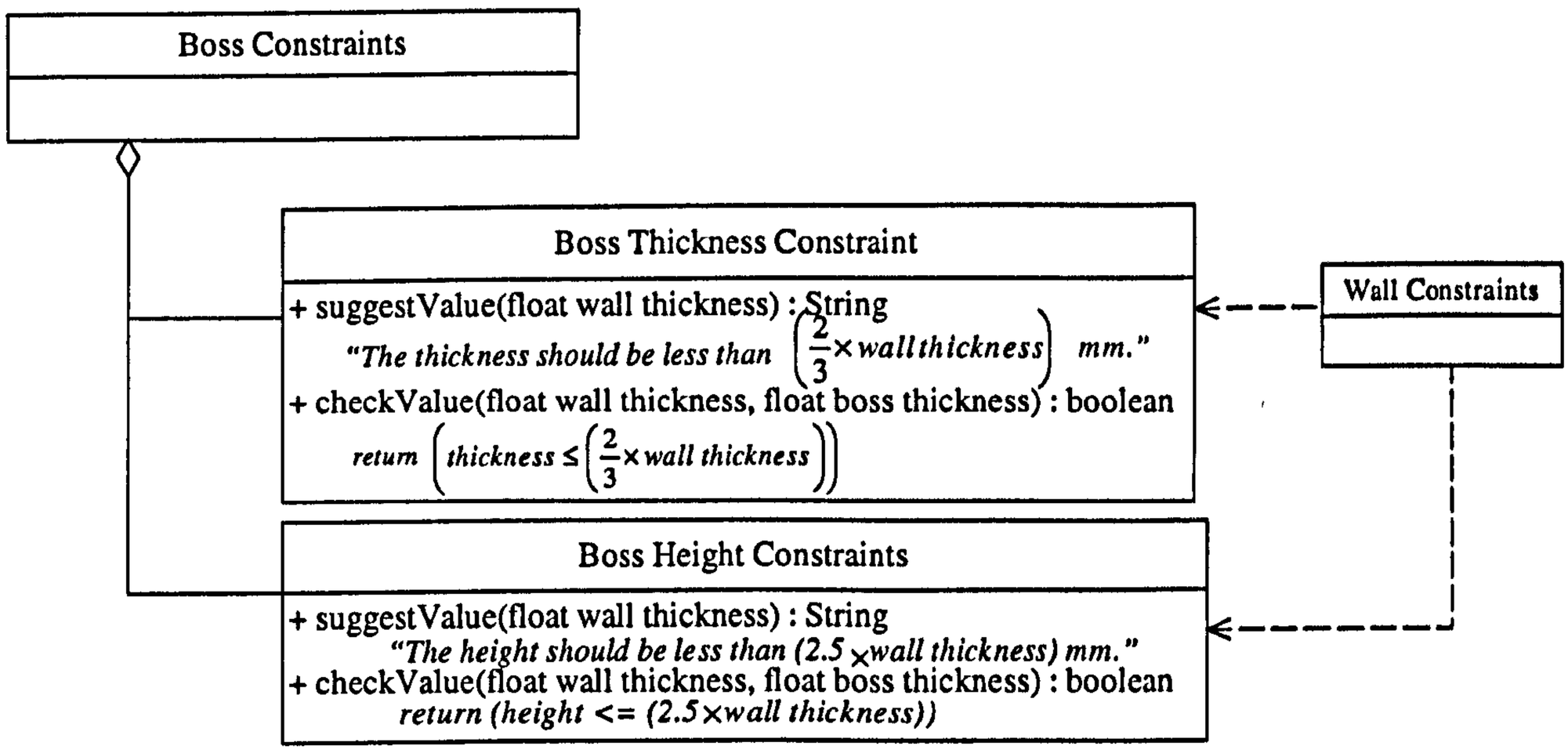


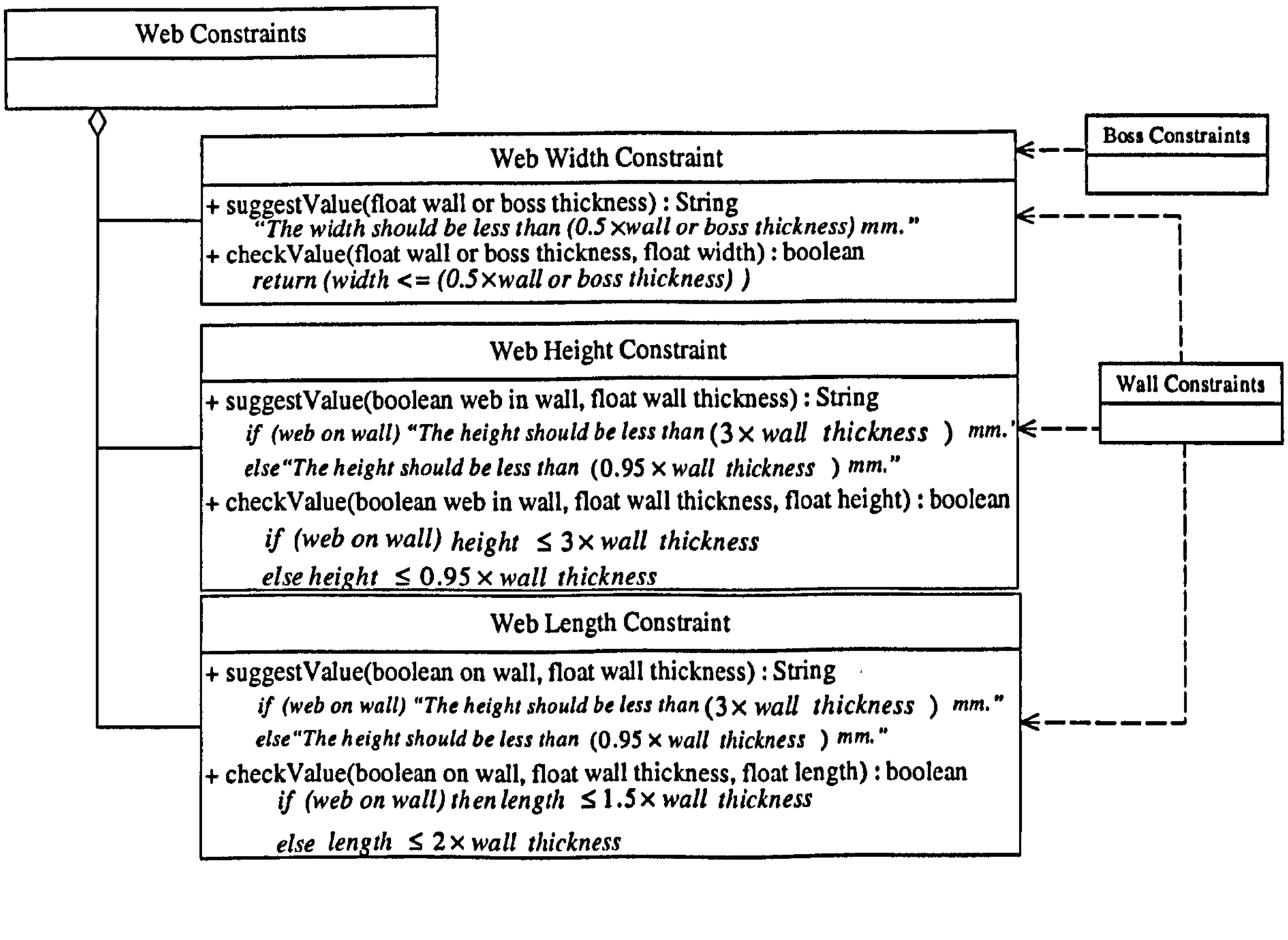
E.2.2 Text constraints



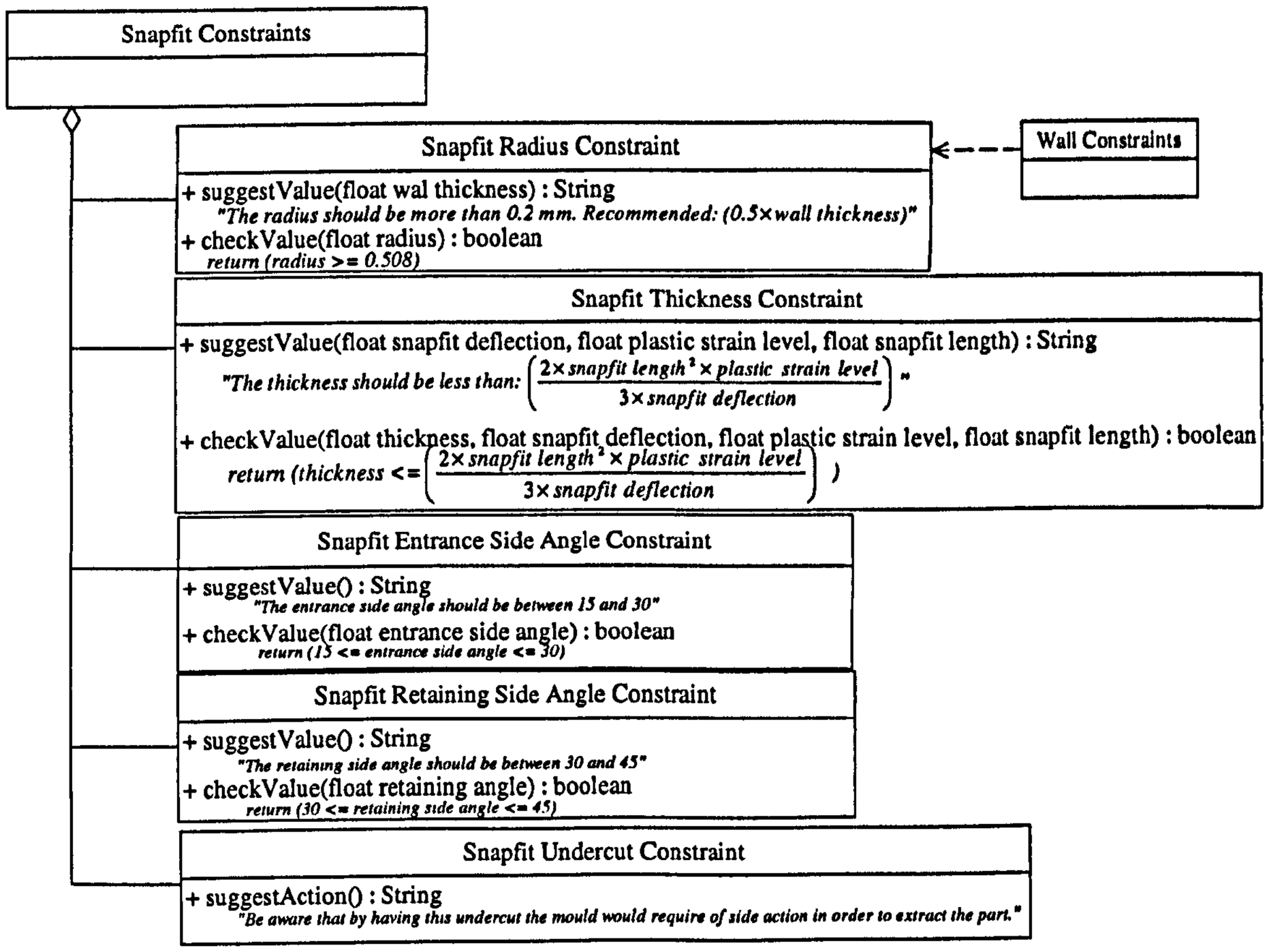
E.2.3 Reinforcement constraints



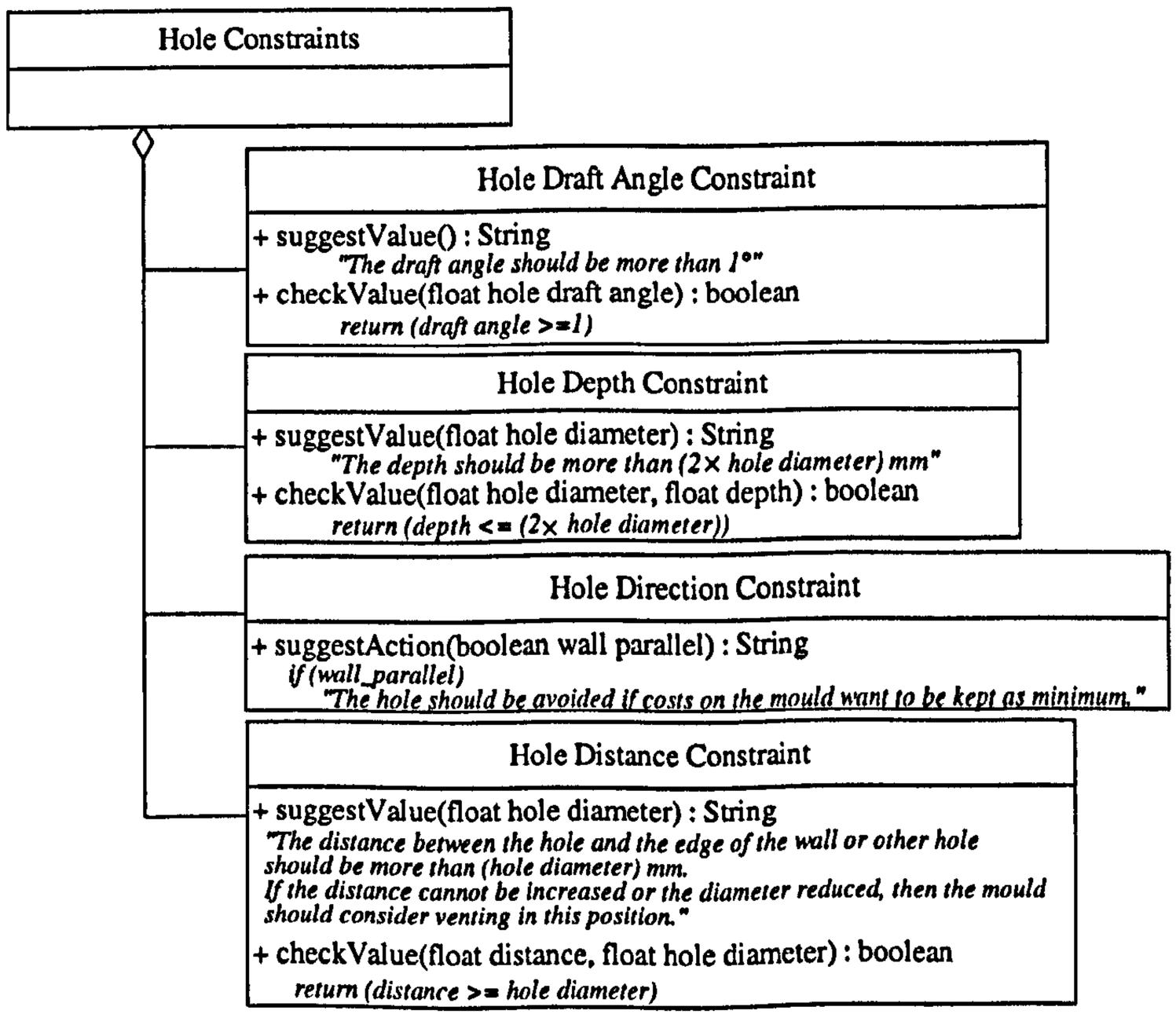

Boss Constraints
Boss Thickness Constraint
+ suggestValue(float wall thickness) : String
"The thickness should be less than $\left(\frac{2}{3}\times wall\,thickness\right)$ mm."
+ checkValue(float wall thickness, float boss thickness) : boolean
return $\left(thickness \le \left(\frac{2}{3}\times wall\ thickness\right)\right)$
Wall Constraints
Boss Height Constraints
+ suggestValue(float wall thickness) : String
"The height should be less than (2.5 ×wall thickness) mm."
+ checkValue(float wall thickness, float boss thickness) : boolean
return (height <= (2.5×wall thickness))


Web Constraints
Boss Constraints
Web Width Constraint
+ suggestValue(float wall or boss thickness) : String
"The width should be less than (0.5 ×wall or boss thickness) mm."
+ checkValue(float wall or boss thickness, float width) : boolean
return (width <= (0.5×wall or boss thickness))
Web Height Constraint
Wall Constraints
+ suggestValue(boolean web in wall, float wall thickness) : String
if (web on wall) "The height should be less than (3× wall thickness) mm.'
else"The height should be less than (0.95 × wall thickness) mm."
+ checkValue(boolean web in wall, float wall thickness, float height) : boolean
if (web on wall) height ≤ 3× wall thickness
else height ≤ 0.95 × wall thickness
Web Length Constraint
+ suggestValue(boolean on wall, float wall thickness) : String
if (web on wall) "The height should be less than (3× wall thickness) mm."
else"The height should be less than (0.95 × wall thickness) mm."
+ checkValue(boolean on wall, float wall thickness, float length) : boolean
if (web on wall) then length ≤ 1.5× wall thickness
else length ≤ 2× wall thickness

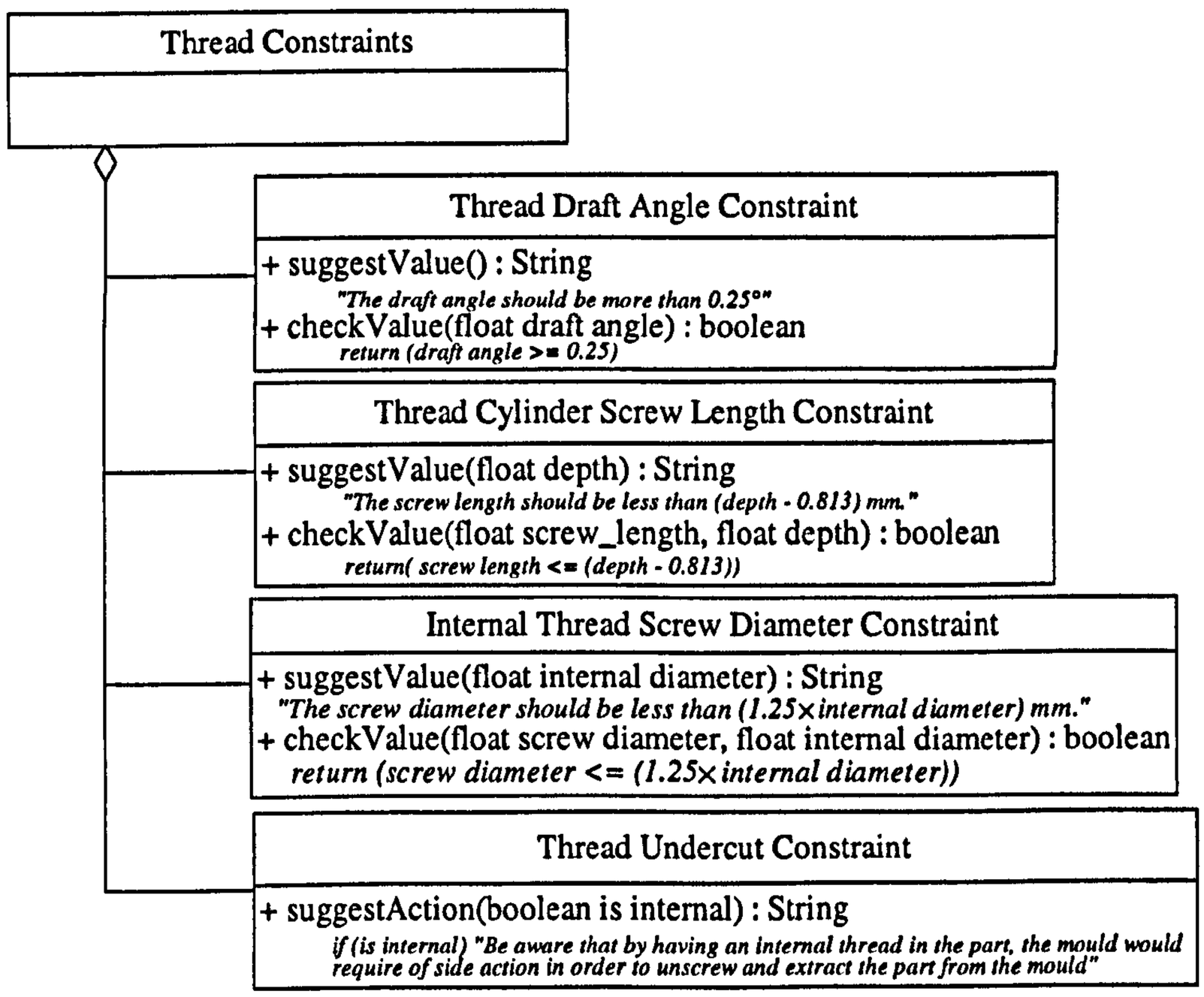
E.2.4 Snapfit constraints



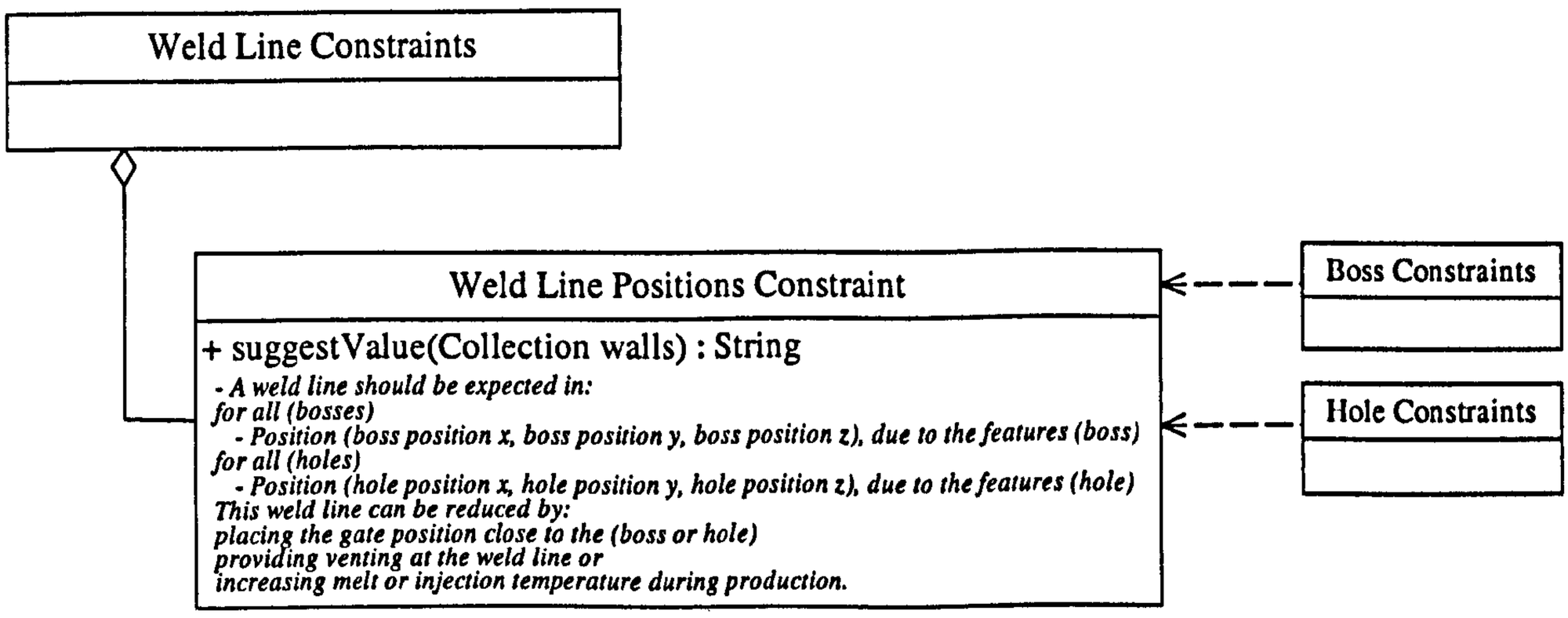
E.2.5 Hole constraints



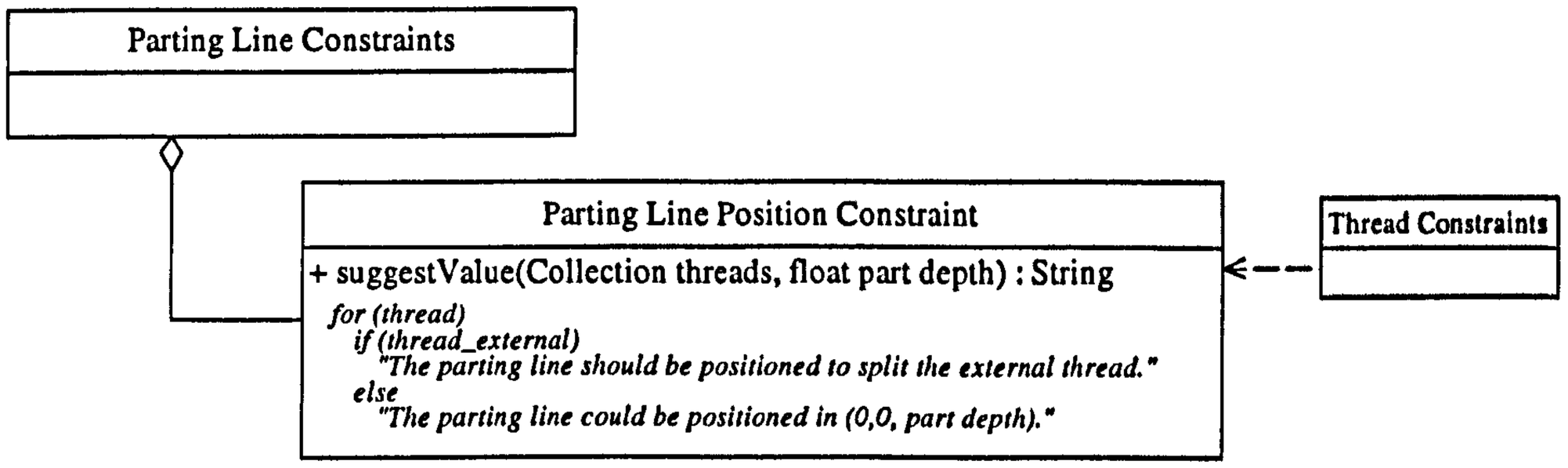
E.2.6 Thread constraints



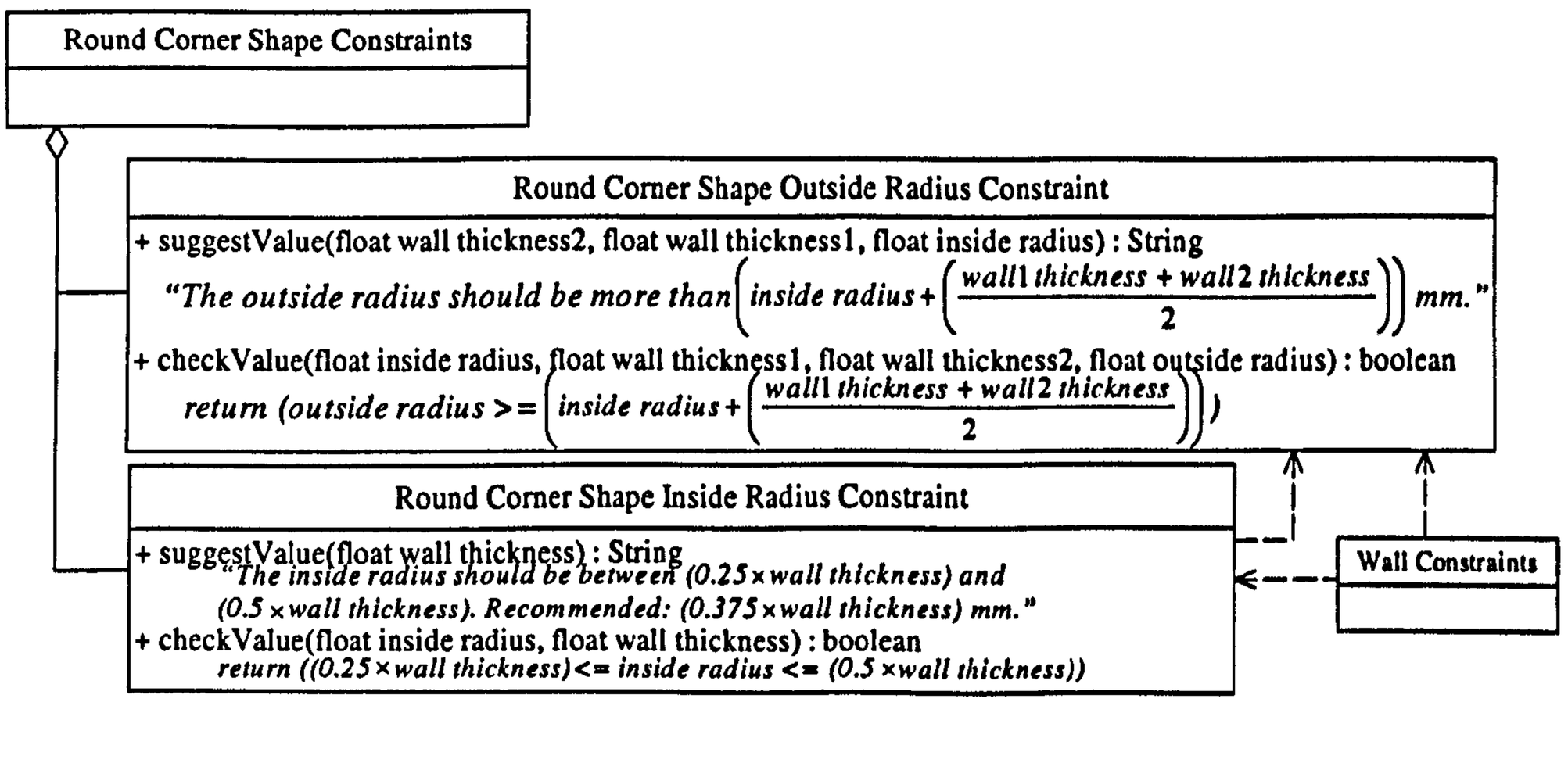
E.2.7 Weld line constraints

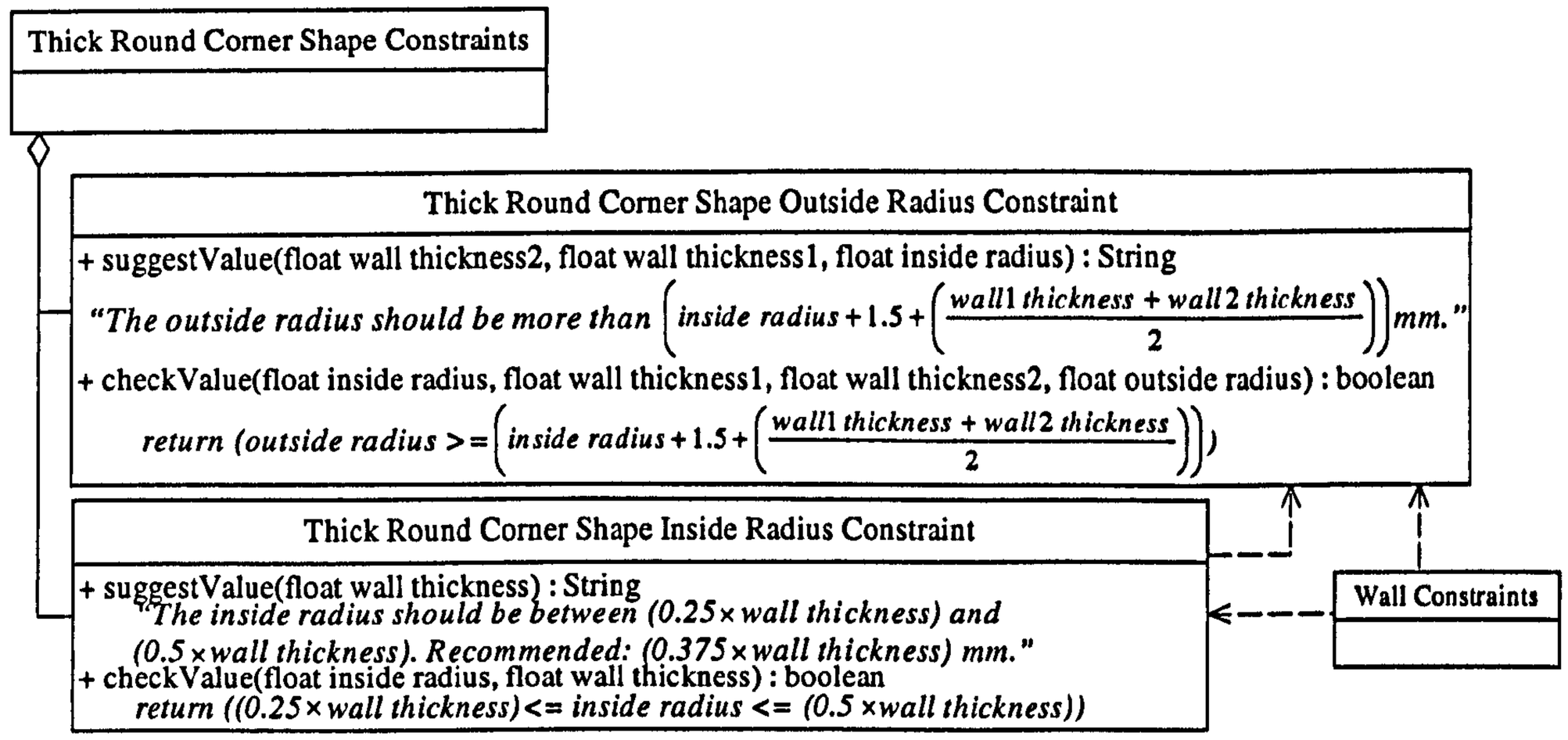


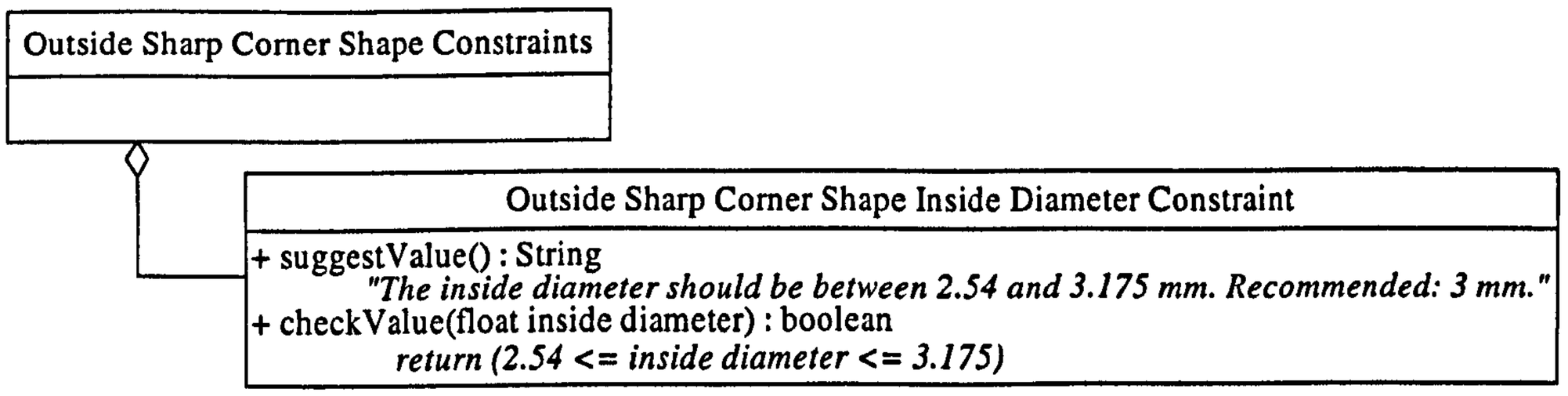
E.2.8 Parting line constraints

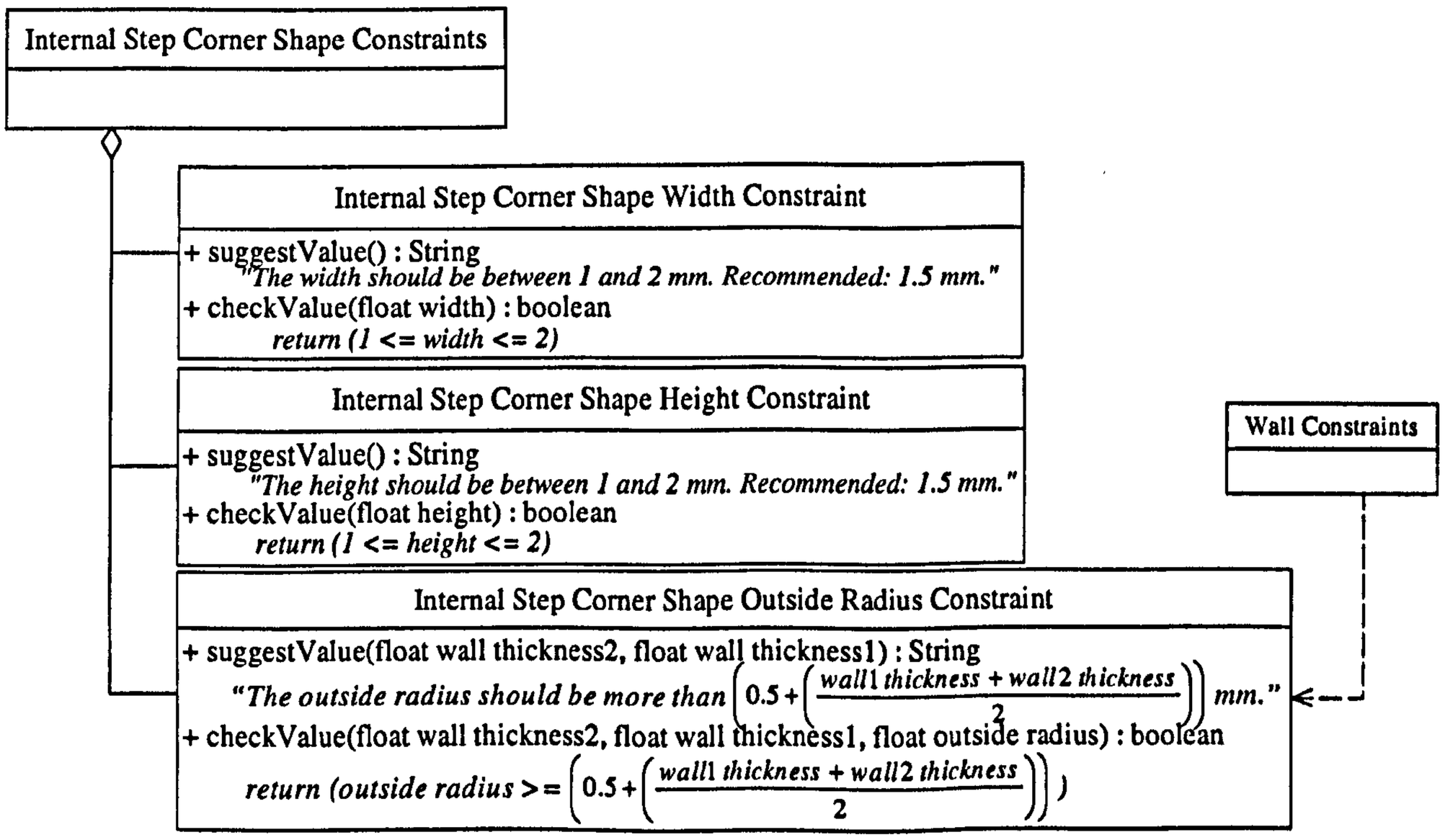


E.2.9 Corner shape constraints

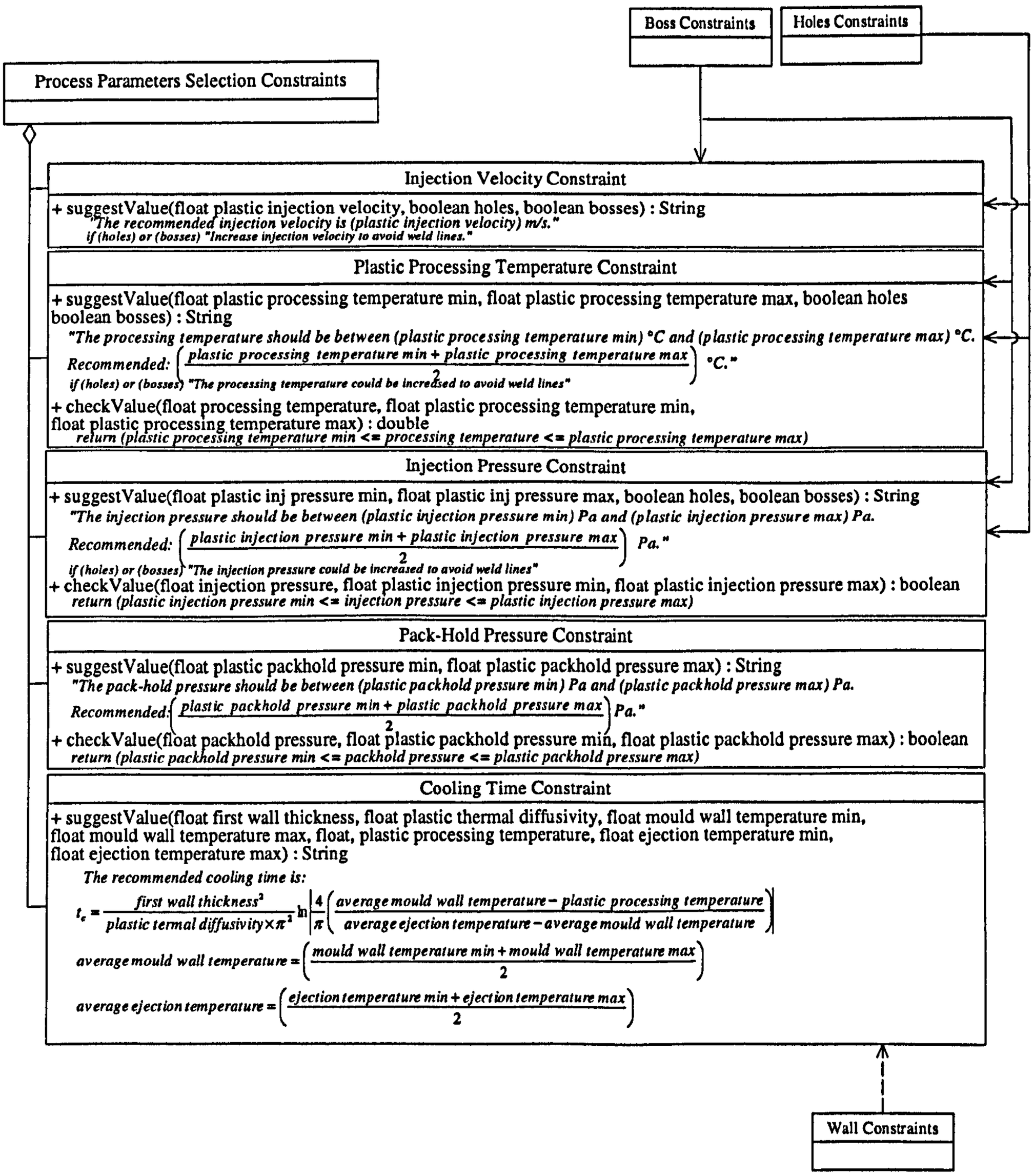



Thick Round Corner Shape Constraints
Thick Round Corner Shape Outside Radius Constraint
+ suggestValue(float wall thickness2, float wall thickness1, float inside radius) : String
"The outside radius should be more than (inside radius + 1.5 + ((wall1 thickness + wall2 thickness)/2)) mm."
+ checkValue(float inside radius, float wall thickness1, float wall thickness2, float outside radius) : boolean
return (outside radius >= (inside radius + 1.5 + ((wall1 thickness + wall2 thickness)/2)))
Thick Round Corner Shape Inside Radius Constraint
+ suggestValue(float wall thickness) : String
"The inside radius should be between (0.25 × wall thickness) and (0.5 × wall thickness). Recommended: (0.375 × wall thickness) mm."
+ checkValue(float inside radius, float wall thickness) : boolean
return ((0.25 × wall thickness) <= inside radius <= (0.5 × wall thickness))
Wall Constraints

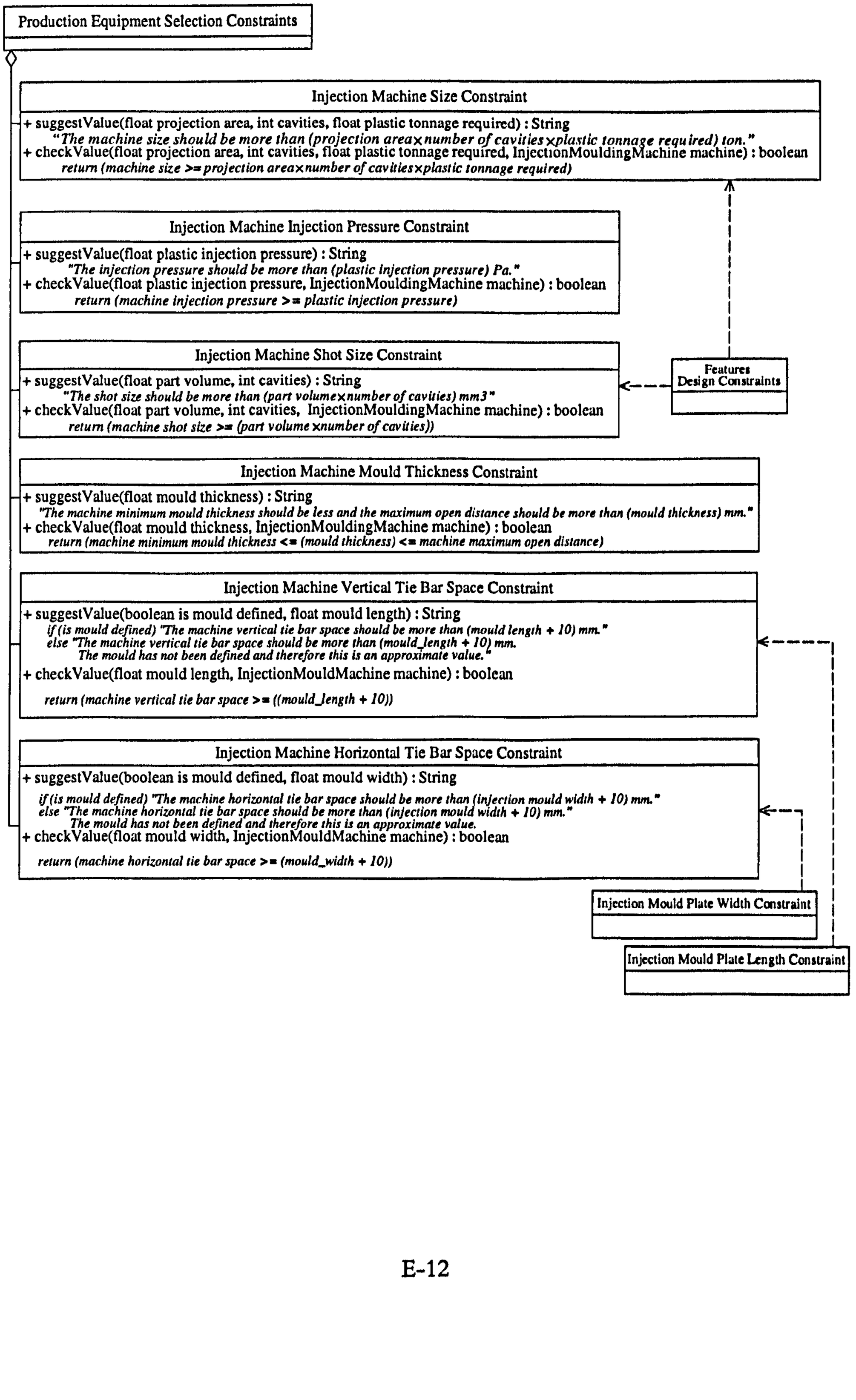

Outside Sharp Corner Shape Constraints
Outside Sharp Corner Shape Inside Diameter Constraint
+ suggestValue() : String
"The inside diameter should be between 2.54 and 3.175 mm. Recommended: 3 mm."
+ checkValue(float inside diameter) : boolean
return (2.54 <= inside diameter <= 3.175)


Internal Step Corner Shape Constraints
Internal Step Corner Shape Width Constraint
+ suggestValue() : String
"The width should be between 1 and 2 mm. Recommended: 1.5 mm."
+ checkValue(float width) : boolean
return (1 <= width <= 2)
Internal Step Corner Shape Height Constraint
+ suggestValue() : String
"The height should be between 1 and 2 mm. Recommended: 1.5 mm."
+ checkValue(float height) : boolean
return (1 <= height <= 2)
Internal Step Corner Shape Outside Radius Constraint
+ suggestValue(float wall thickness2, float wall thickness1) : String
"The outside radius should be more than (0.5 + ((wall1 thickness + wall2 thickness)/2)) mm."
+ checkValue(float wall thickness2, float wall thickness1, float outside radius) : boolean
return (outside radius >= (0.5 + ((wall1 thickness + wall2 thickness)/2)))
Wall Constraints

E.3 Process parameters selection constraints



E.4 Production equipment selection constraints



E.5 Mould design constraints

Injection Mould Design Constraints

+ calculateS(float part length or width or depth) : double
if (0 <= part width or length <= 25) s=5
if (26 <= part width or length <= 50) s=10
if (51 <= part width or length <= 75) s=20
if (76 <= part width or length <= 100) s=30
if (101 <= part width or length <= 125) s=40
if (126 <= part width or length <= 150) s=55
if (151 <= part width or length <= 175) s=75
if (176 <= part width or length <= 200) s=100
if (201 <= part width or length <= 225) s=125
if (226 <= part width or length <= 250) s=150

Injection Mould Plate Length Constraint

+ suggestValue(String cav arrangement, int cavities, float part width, float part length) : double
if (cav arrangement = unbalanced 1)
if (part width > part length)
calculateS()
$plate length = \left(\left(\frac{num cavities}{2}+2\right)\times s\right)+\left(\frac{num cavities}{2}\times part\ width\right)$
else
calculateS()
$plate length = \left(\left(\frac{num cavities}{2}+2\right)\times s\right)+\left(\frac{num cavities}{2}\times part\ length\right)$
if (cav arrangement = unbalanced 2)
calculateS()
$plate length = \left(\left(\frac{num cavities}{4}+2\right)\times s\right)+\left(\frac{num cavities}{4}\times part\ length\right)$
if (cav arrangement = balanced 1)
calculateS()
$plate length = (4\times s)+(2\times part\ length)$
if (cav arrangement = balanced 2)
if (part width > part length)
calculateS()
$plate length = (4\times s)+(2\times part\ width)$
else
calculateS()
$plate length = (4\times s)+(2\times part\ length)$

Injection Mould Plate Width Constraint

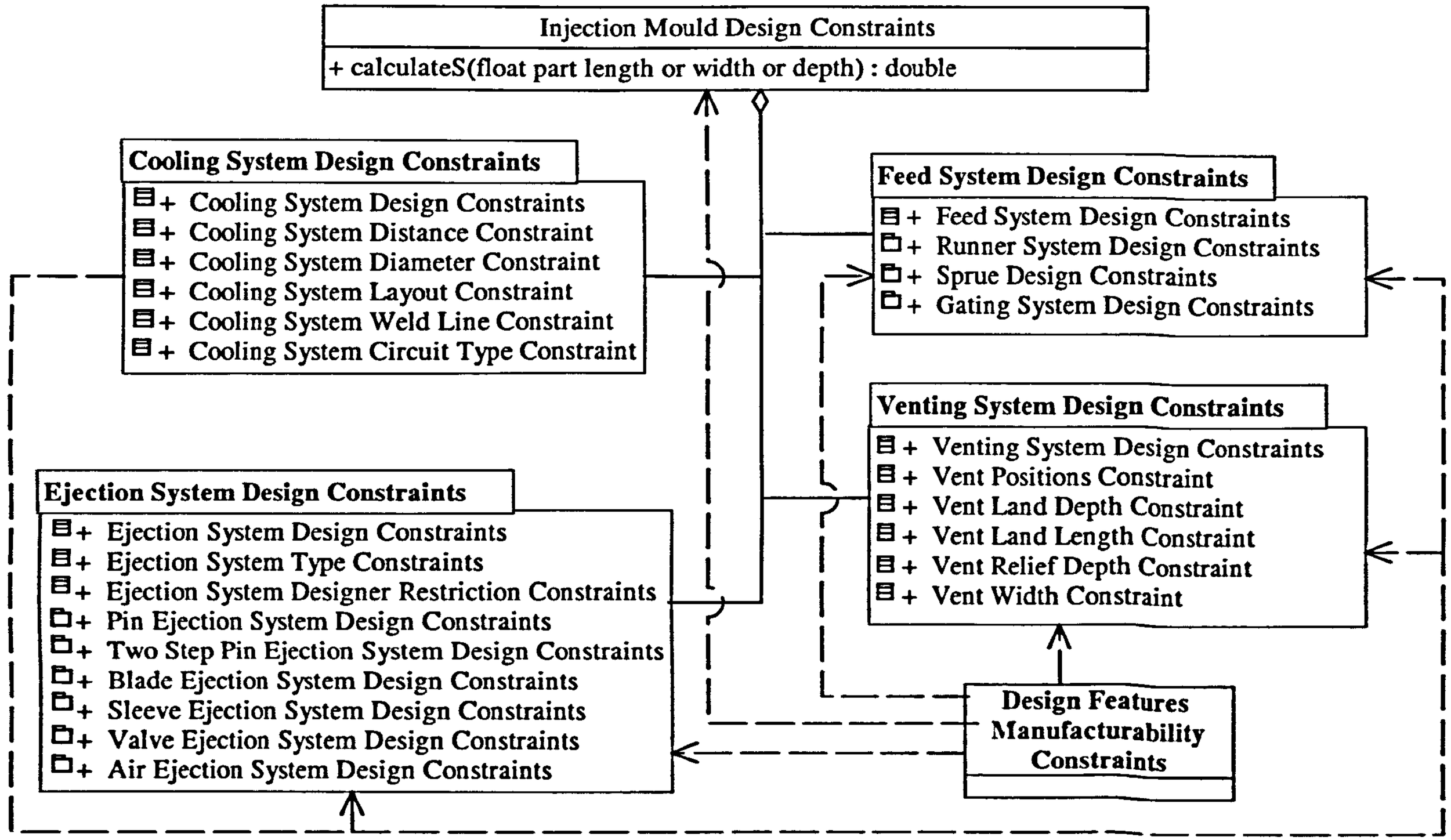
+ suggestValue(String cav arrangement, int cavities, float part width, float part length) : double
if (cav arrangement = unbalanced 1)
if (part width > part length)
calculateS()
$plate width = (4\times s)+(2\times part\ length)$
else
calculateS()
$plate width = (4\times s)+(2\times part\ width)$
if (cav arrangement = unbalanced 2)
calculateS()
$plate width = \left(\left(\frac{cavities}{4}+2\right)\times s\right)+\left(\frac{cavities}{4}\times part\ width\right)$
if (cav arrangement = balanced 1)
calculateS()
$plate width = (4\times s)+(2\times part\ width)$
if (cav arrangement = balanced 2)
if (part width > part length)
calculateS()
$plate width = \left(\left(\frac{num cavities}{4}+1\right)\times 2\times s\right)+\left(\frac{num cavities}{2}\times part\ length\right)$
else
calculateS()
$plate width = \left(\left(\frac{num cavities}{4}+1\right)\times 2\times s\right)+\left(\frac{num cavities}{2}\times part\ width\right)$

Injection Mould Thickness Constraint

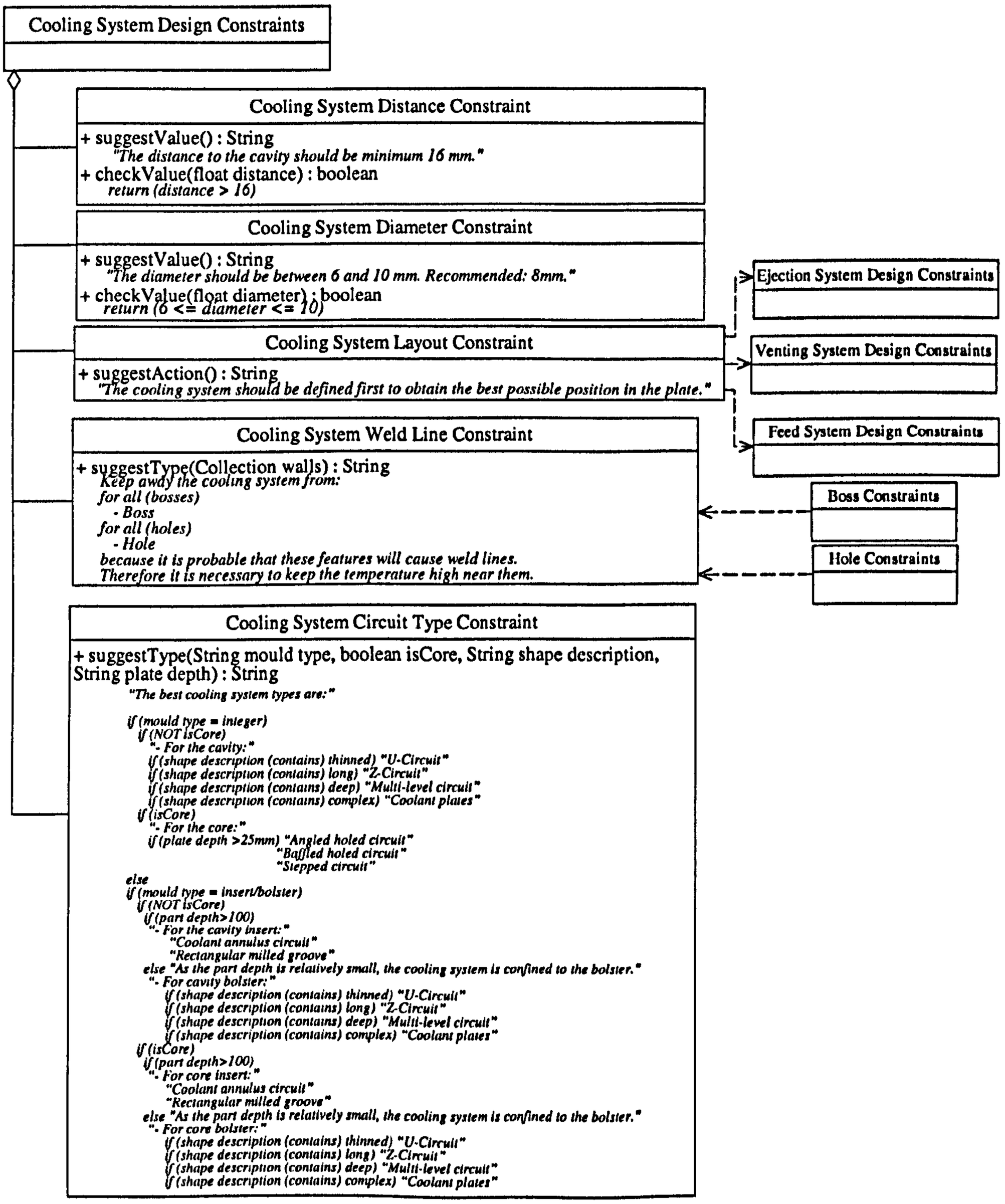
+ suggestValue(String cav arrangement, float part depth) : double
calculateS()
$mould\ thickness = (2\times s)+product\ depth$

Standard Mould Constraint

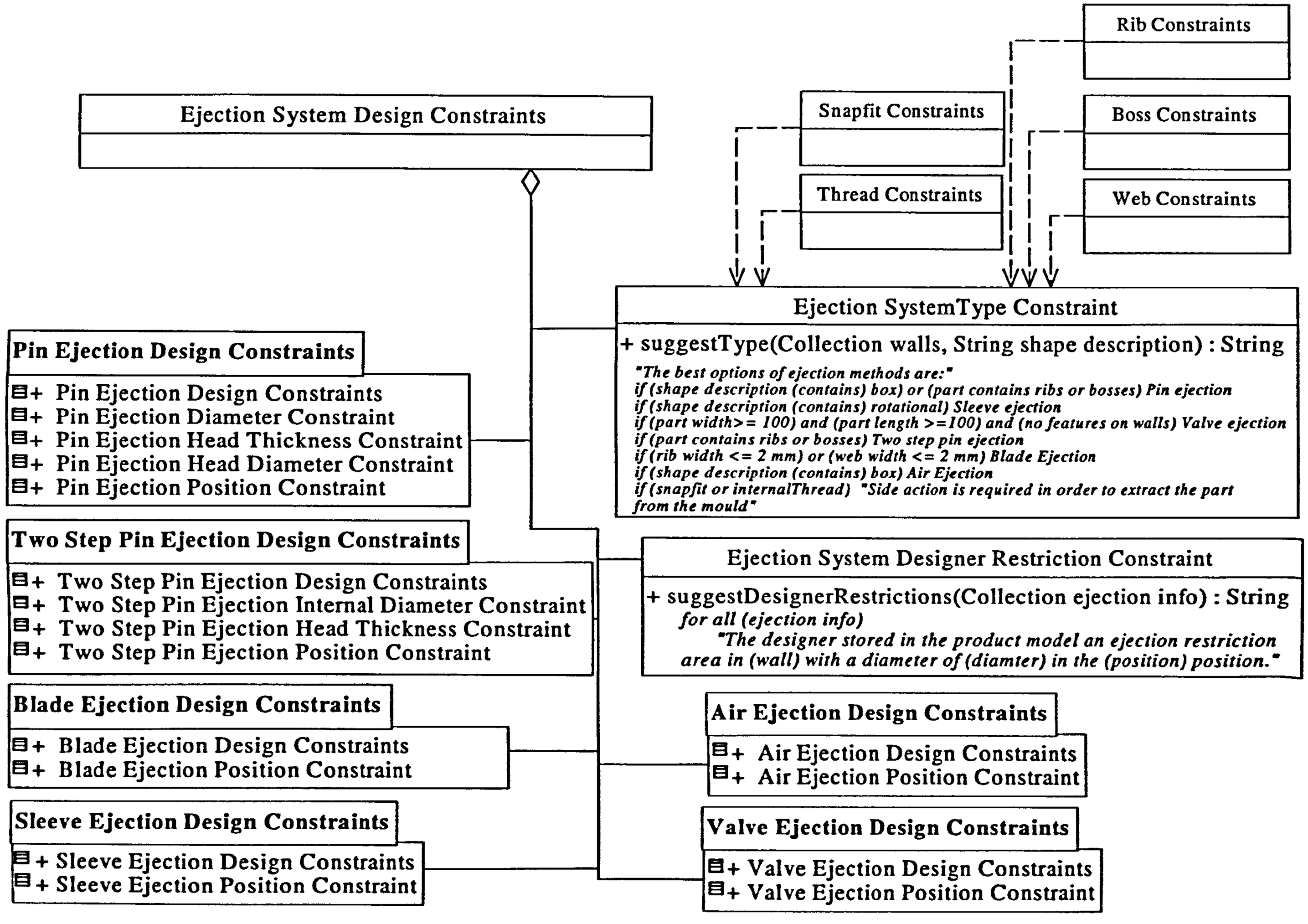
+ suggestStdMould(Collection std moulds, float plate length, float plate width, float mould thickness) : String
for all (std moulds)
if (length suggested >= std mould length >= part length) and
(width suggested >= std mould width >= part width) and
$\left(thickness\ suggested >= \frac{std\ mould\ min\ thickness + std\ mould\ max\ thickness}{2} >= mould\ thickness\right)$
std mould = std mould name
length suggested = std mould length
width suggested = std mould width
$thickness\ suggested = \frac{std\ mould\ min\ thickness + std\ mould\ max\ thickness}{2}$
"The recommended standard mould is (std mould name) with a length of (length),
width of (width) and an average thickness of (thickness suggested)"

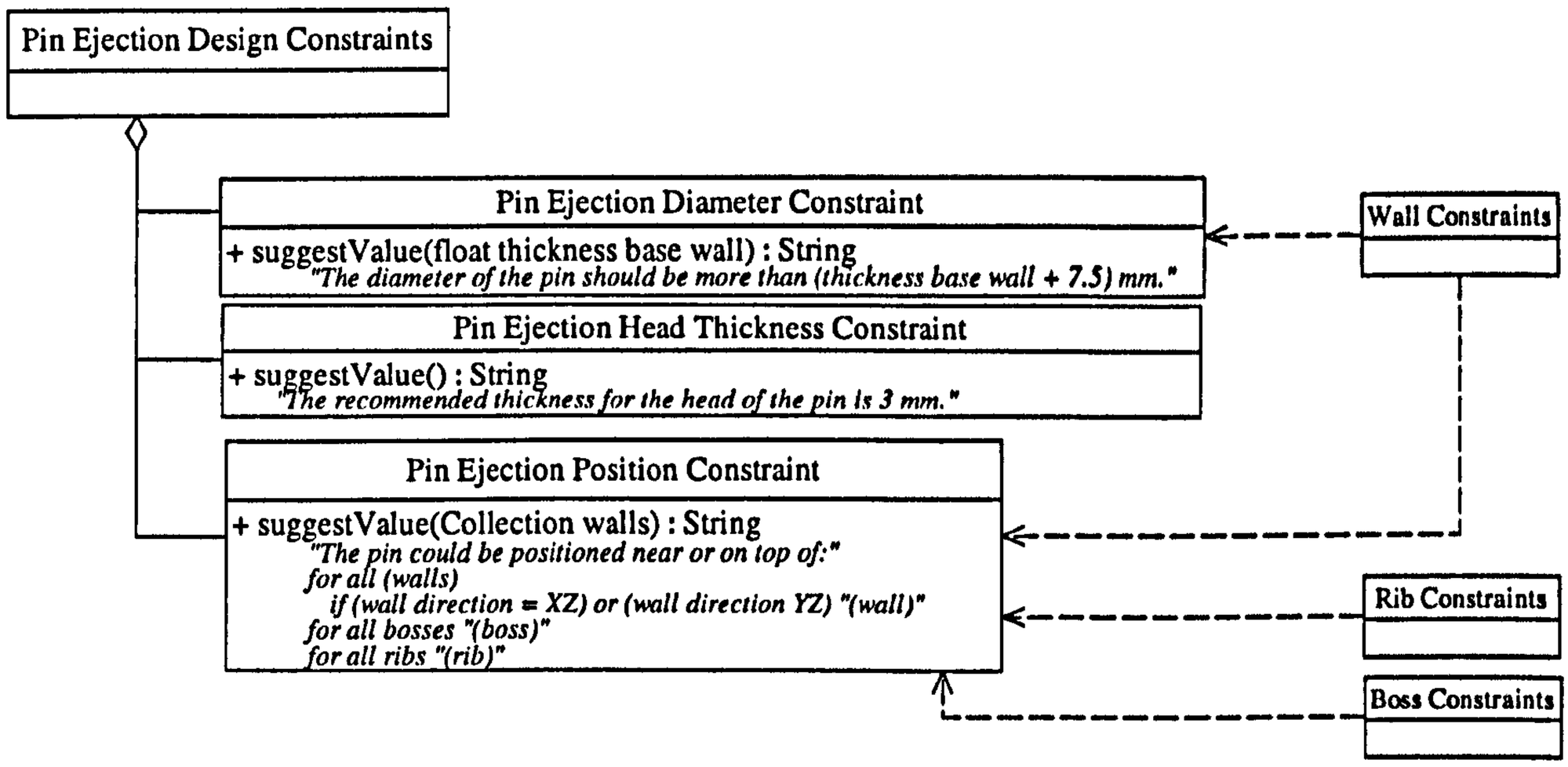

Injection Mould Design Constraints
+ calculateS(float part length or width or depth) : double
Cooling System Design Constraints
+ Cooling System Design Constraints
+ Cooling System Distance Constraint
+ Cooling System Diameter Constraint
+ Cooling System Layout Constraint
+ Cooling System Weld Line Constraint
+ Cooling System Circuit Type Constraint
Feed System Design Constraints
+ Feed System Design Constraints
+ Runner System Design Constraints
+ Sprue Design Constraints
+ Gating System Design Constraints
Venting System Design Constraints
+ Venting System Design Constraints
+ Vent Positions Constraint
+ Vent Land Depth Constraint
+ Vent Land Length Constraint
+ Vent Relief Depth Constraint
+ Vent Width Constraint
Ejection System Design Constraints
+ Ejection System Design Constraints
+ Ejection System Type Constraints
+ Ejection System Designer Restriction Constraints
+ Pin Ejection System Design Constraints
+ Two Step Pin Ejection System Design Constraints
+ Blade Ejection System Design Constraints
+ Sleeve Ejection System Design Constraints
+ Valve Ejection System Design Constraints
+ Air Ejection System Design Constraints
Design Features Manufacturability Constraints

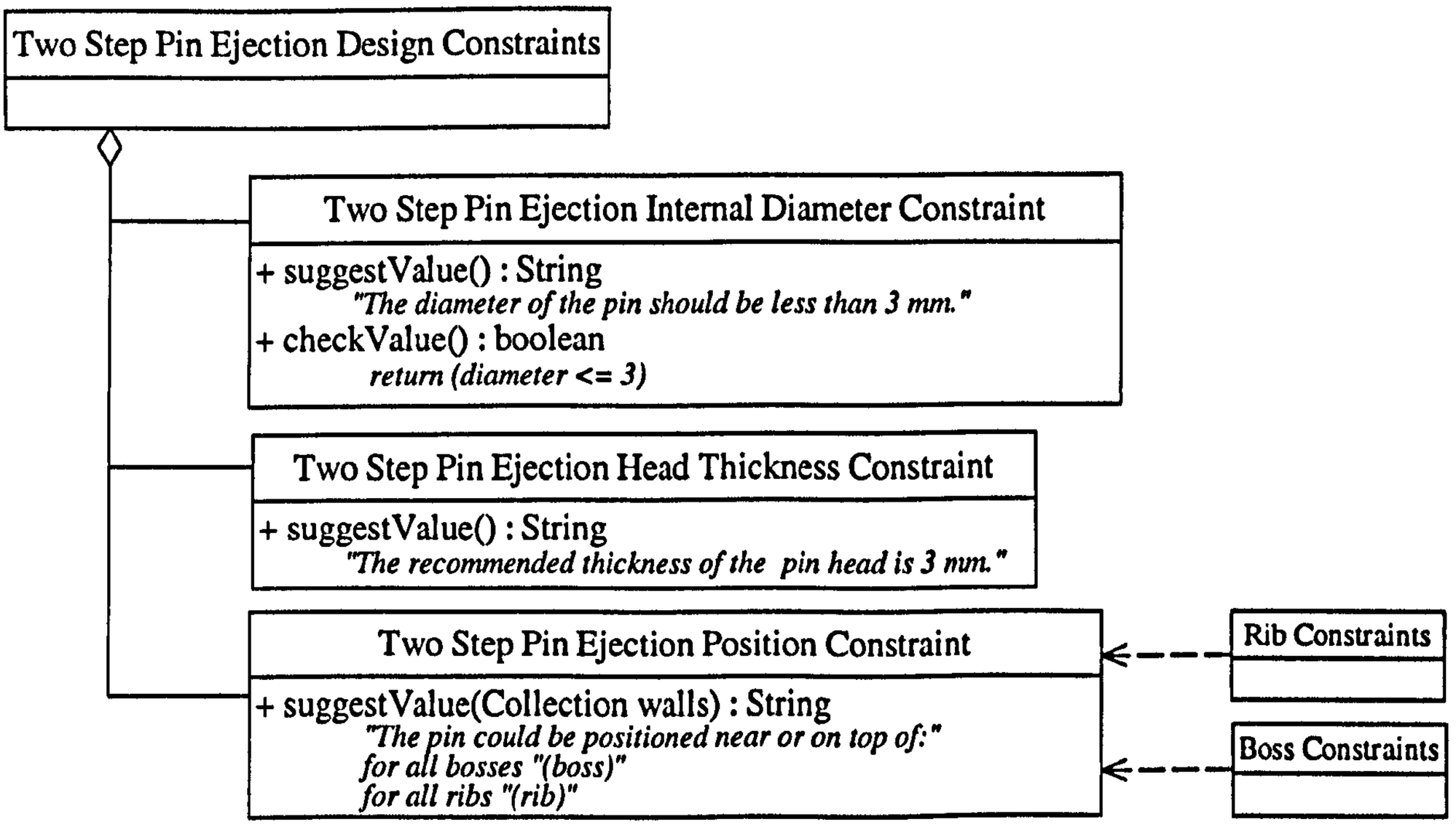
E.5.1 Cooling system design constraints

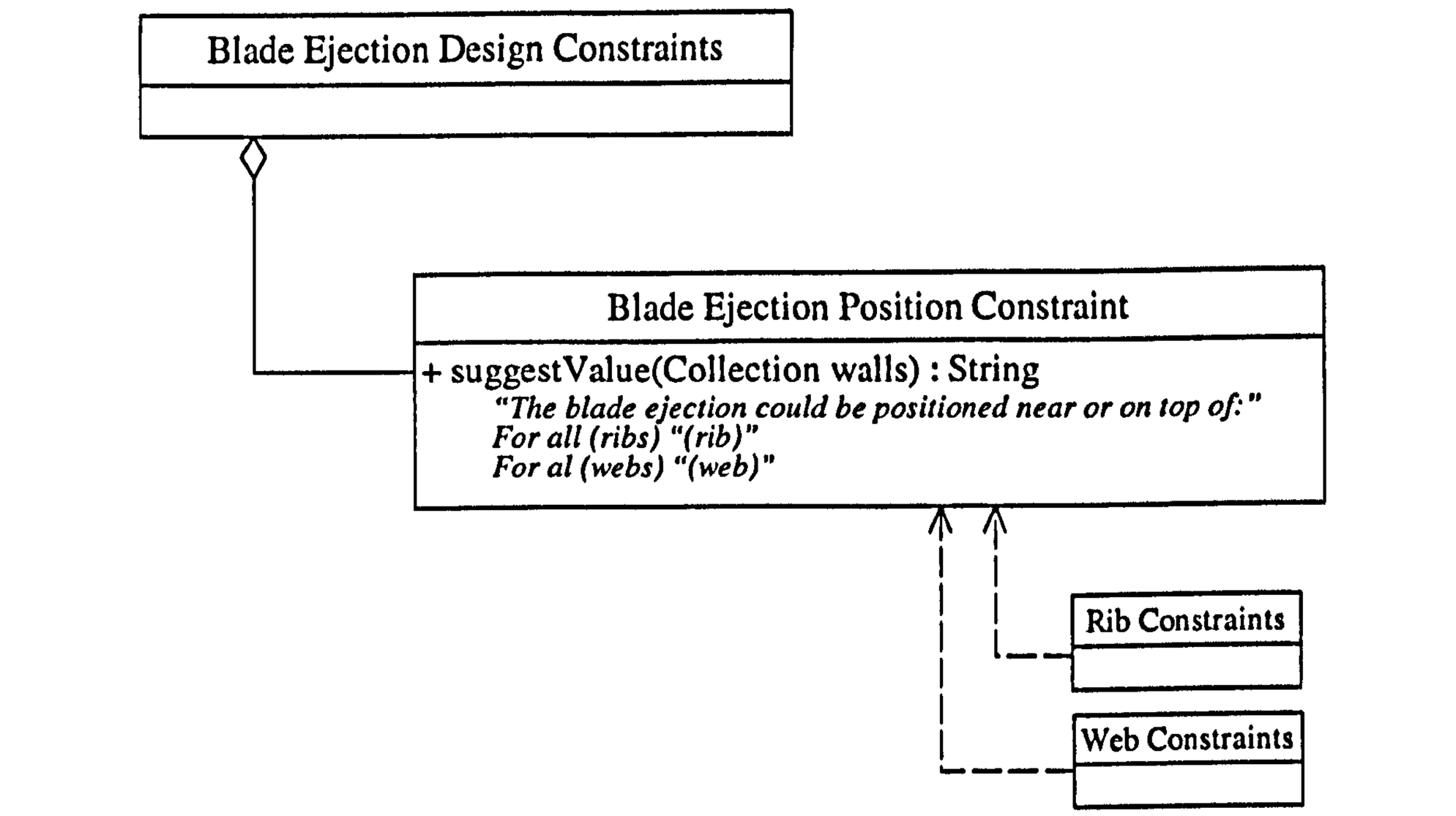


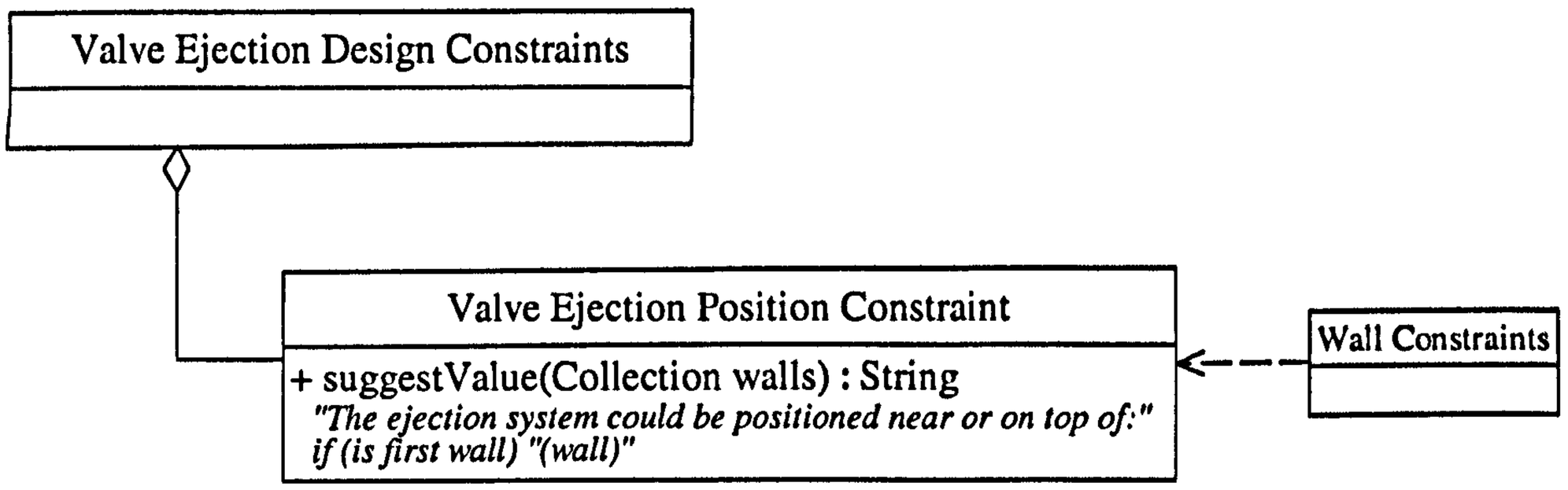
E.5.2 Ejection system design constraints



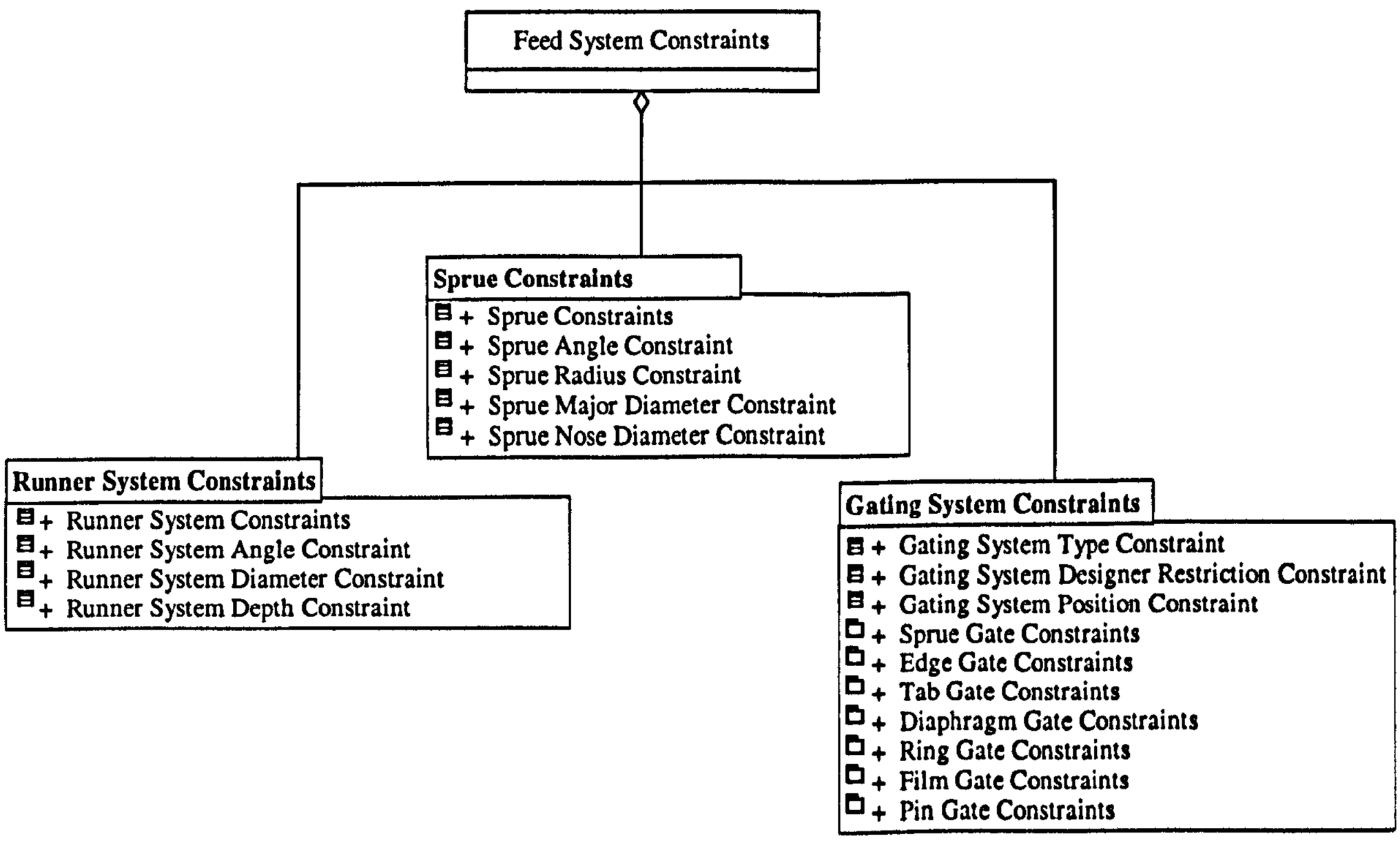

Pin Ejection Design Constraints
Pin Ejection Diameter Constraint
+ suggestValue(float thickness base wall) : String
"The diameter of the pin should be more than (thickness base wall + 7.5) mm."
Pin Ejection Head Thickness Constraint
+ suggestValue() : String
"The recommended thickness for the head of the pin is 3 mm."
Pin Ejection Position Constraint
+ suggestValue(Collection walls) : String
"The pin could be positioned near or on top of:"
for all (walls)
if (wall direction = XZ) or (wall direction YZ) "(wall)"
for all bosses "(boss)"
for all ribs "(rib)"
Wall Constraints
Rib Constraints
Boss Constraints


Two Step Pin Ejection Design Constraints
Two Step Pin Ejection Internal Diameter Constraint
+ suggestValue() : String
"The diameter of the pin should be less than 3 mm."
+ checkValue() : boolean
return (diameter <= 3)
Two Step Pin Ejection Head Thickness Constraint
+ suggestValue() : String
"The recommended thickness of the pin head is 3 mm."
Two Step Pin Ejection Position Constraint
+ suggestValue(Collection walls) : String
"The pin could be positioned near or on top of:"
for all bosses "(boss)"
for all ribs "(rib)"
Rib Constraints
Boss Constraints

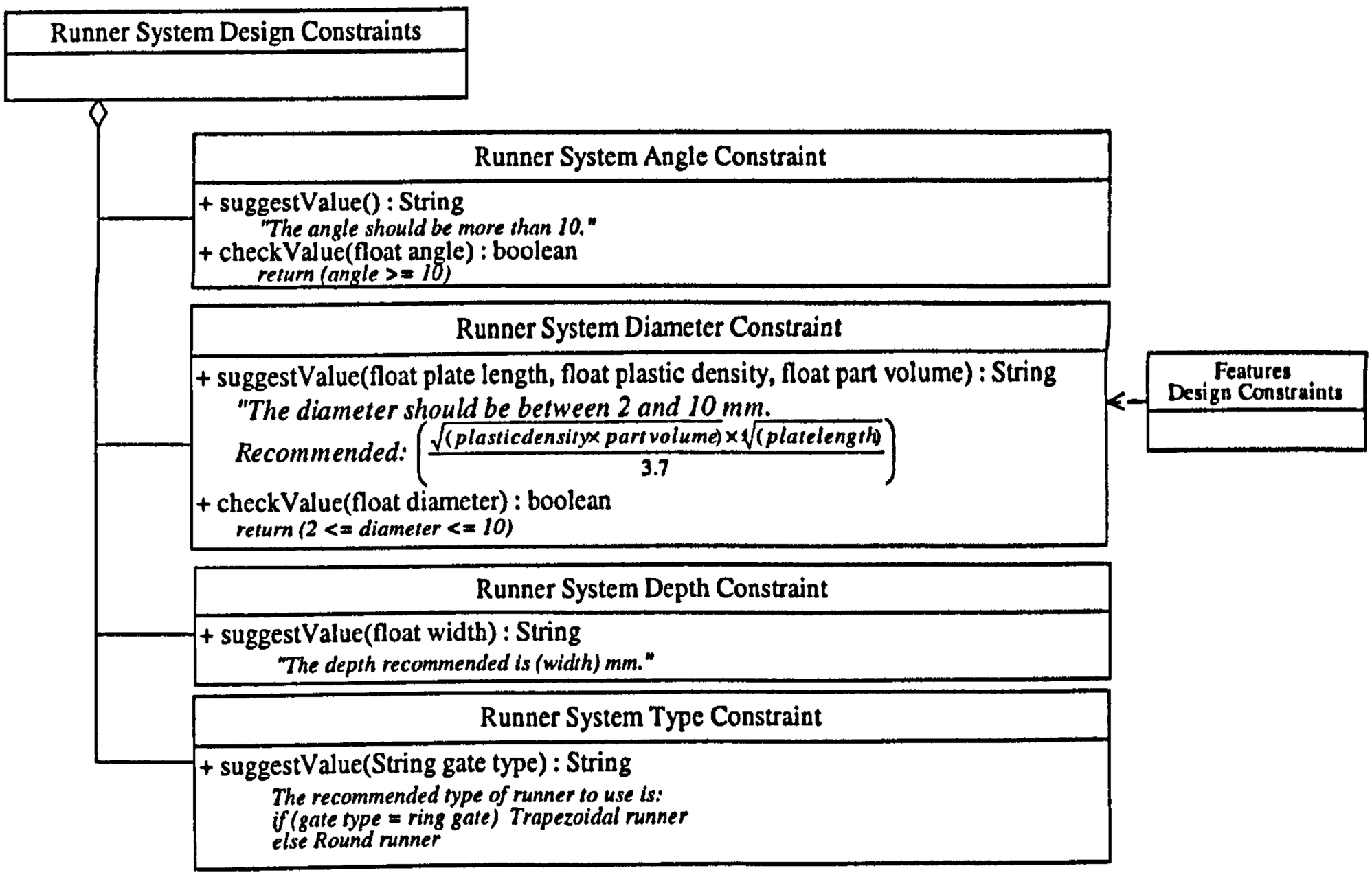

Blade Ejection Design Constraints
Blade Ejection Position Constraint
+ suggestValue(Collection walls) : String
"The blade ejection could be positioned near or on top of:"
For all (ribs) "(rib)"
For al (webs) "(web)"
Rib Constraints
Web Constraints
Sleeve Ejection Design Constraints
Sleeve Ejection Positions Constraint
+ suggestValue(Collection walls) : String
"The sleeve could be positioned near or on top of:"
for all (rotational walls)
if (wall type = cylinder) "(wall)"
Wall Constraints
Air Ejection System Design Constraints
Air Ejection Position Constraint
+ suggestValue(Collection walls) : String
"The ejection system could be positioned near or on top of:"
if (is first wall) "(wall)"
Wall Constraints



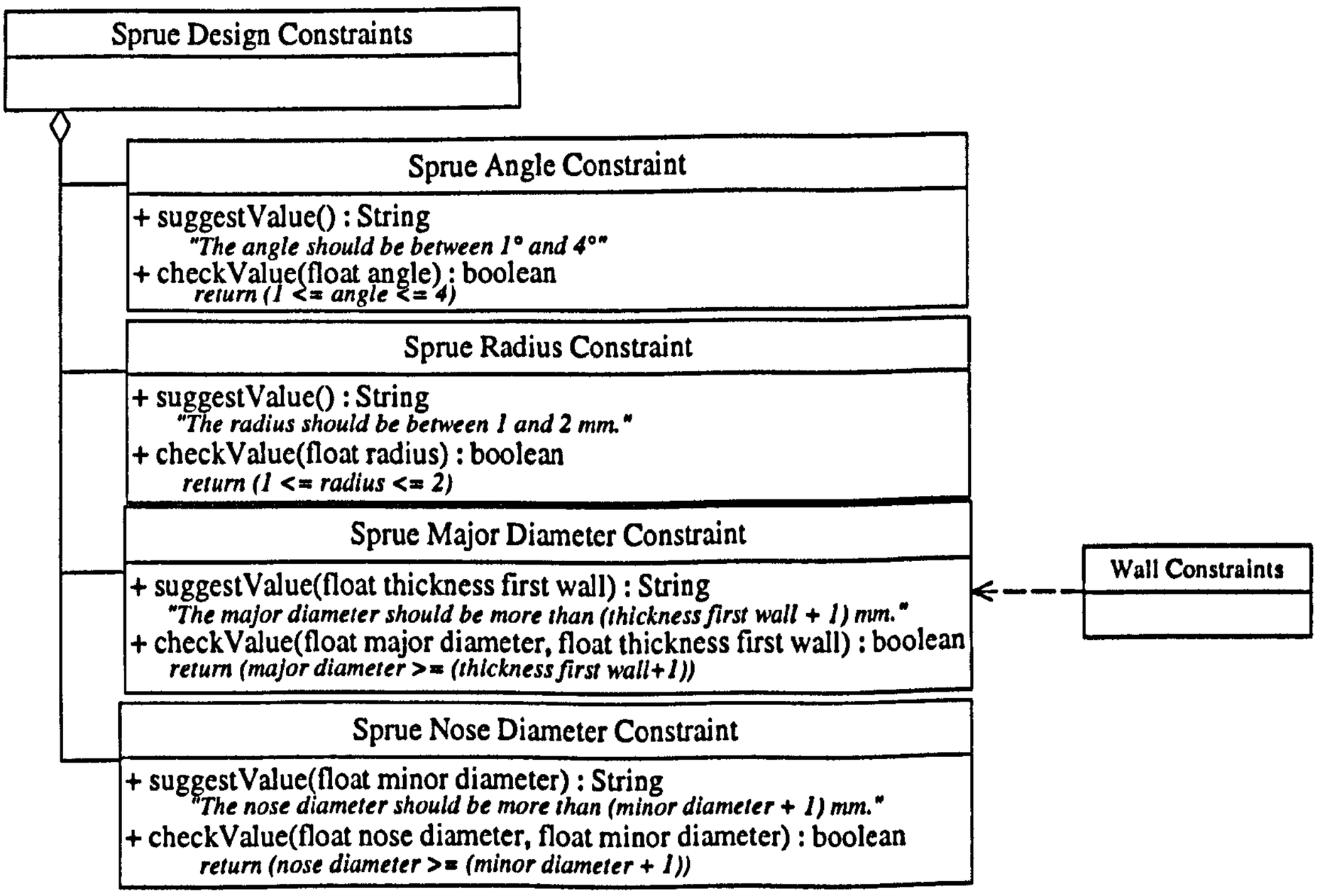
E.5.3 Feed system design constraints



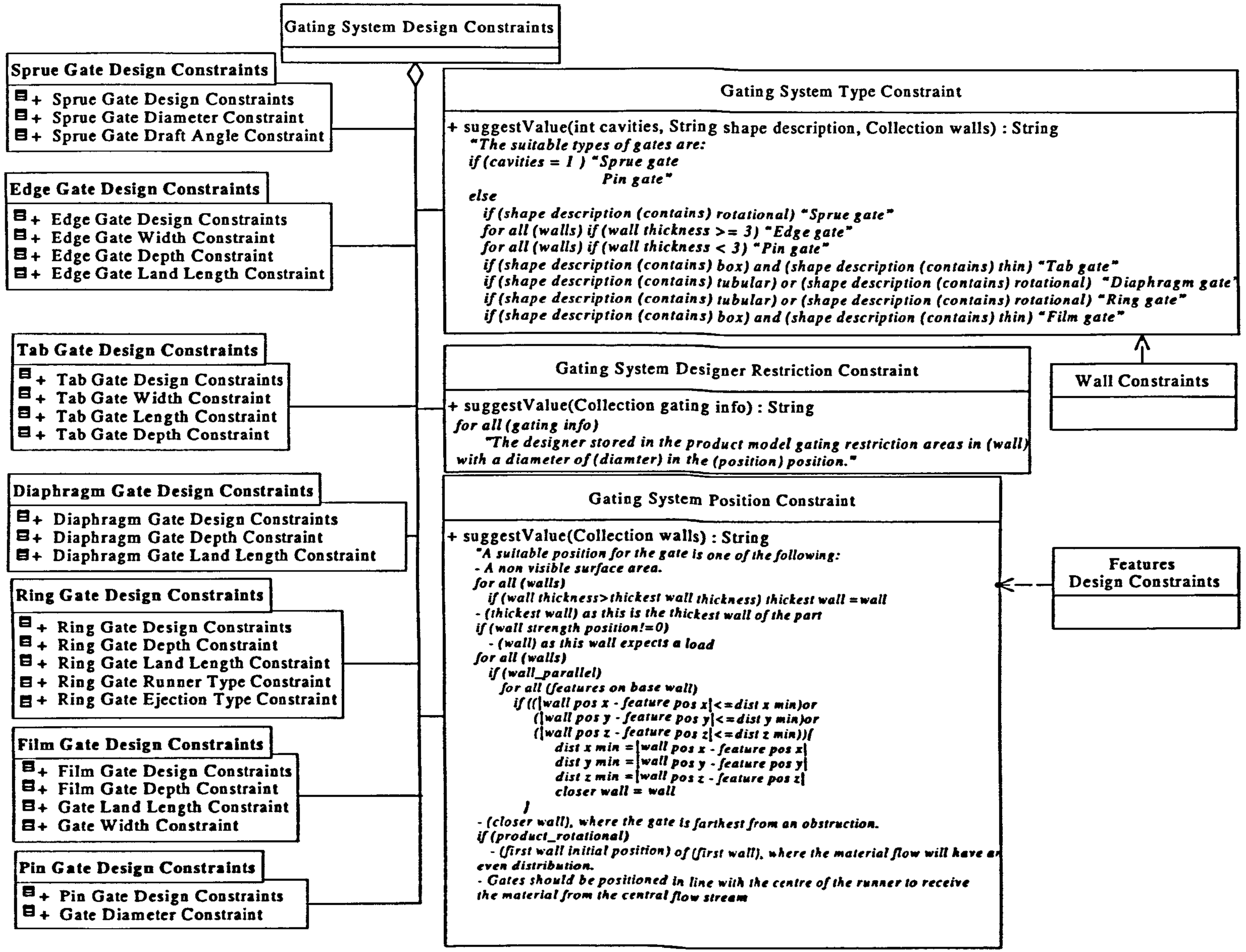
E.5.3.1 Runner system design constraints

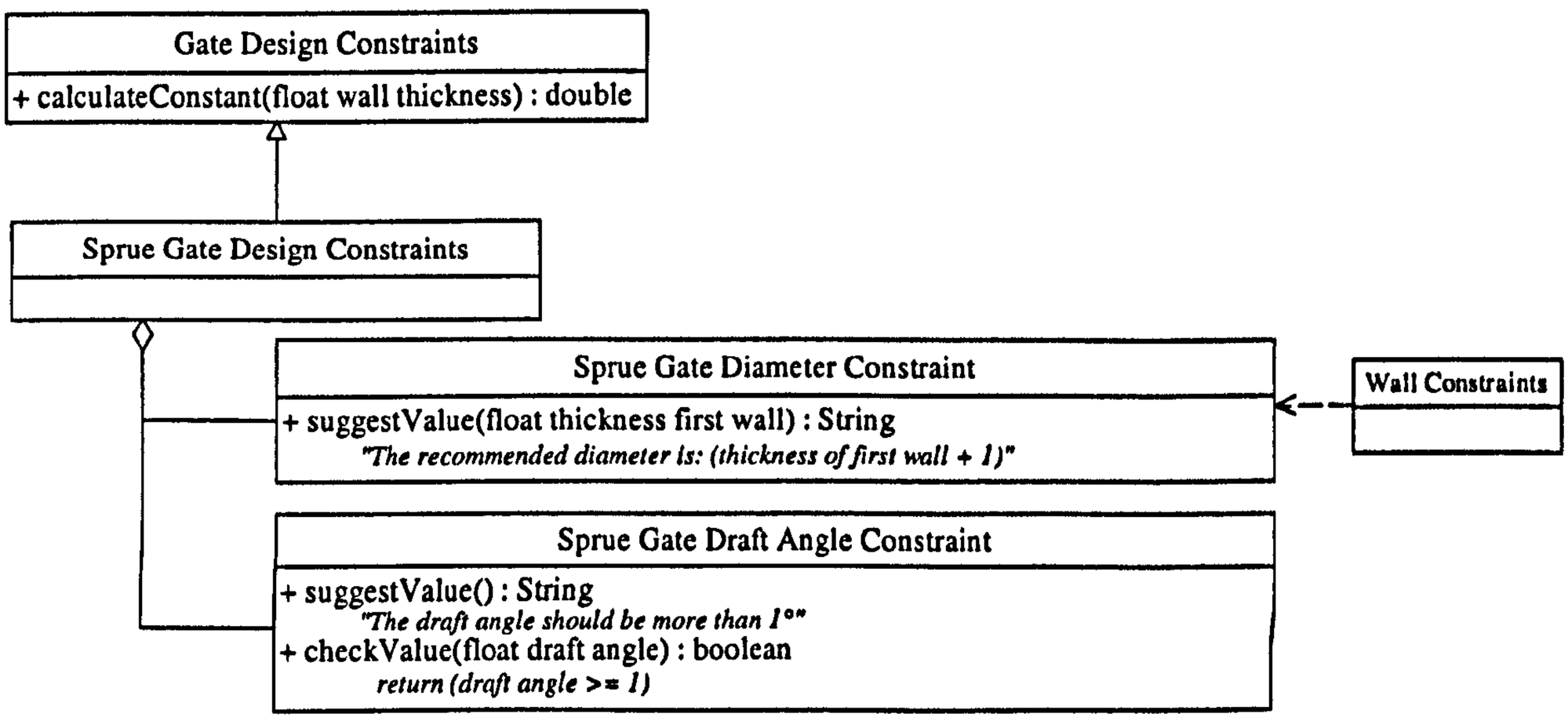


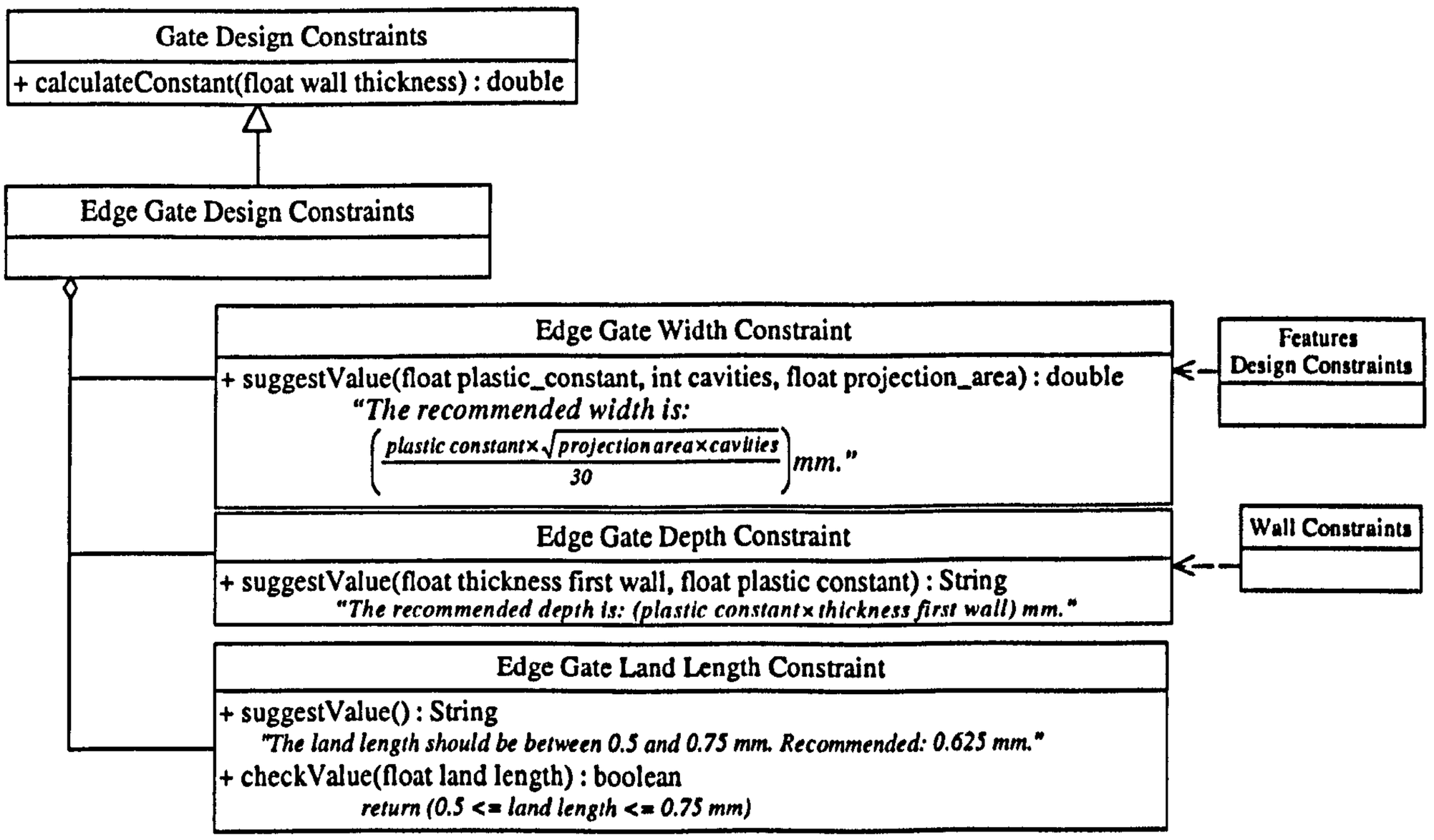
E.5.3.2 Sprue system design constraints

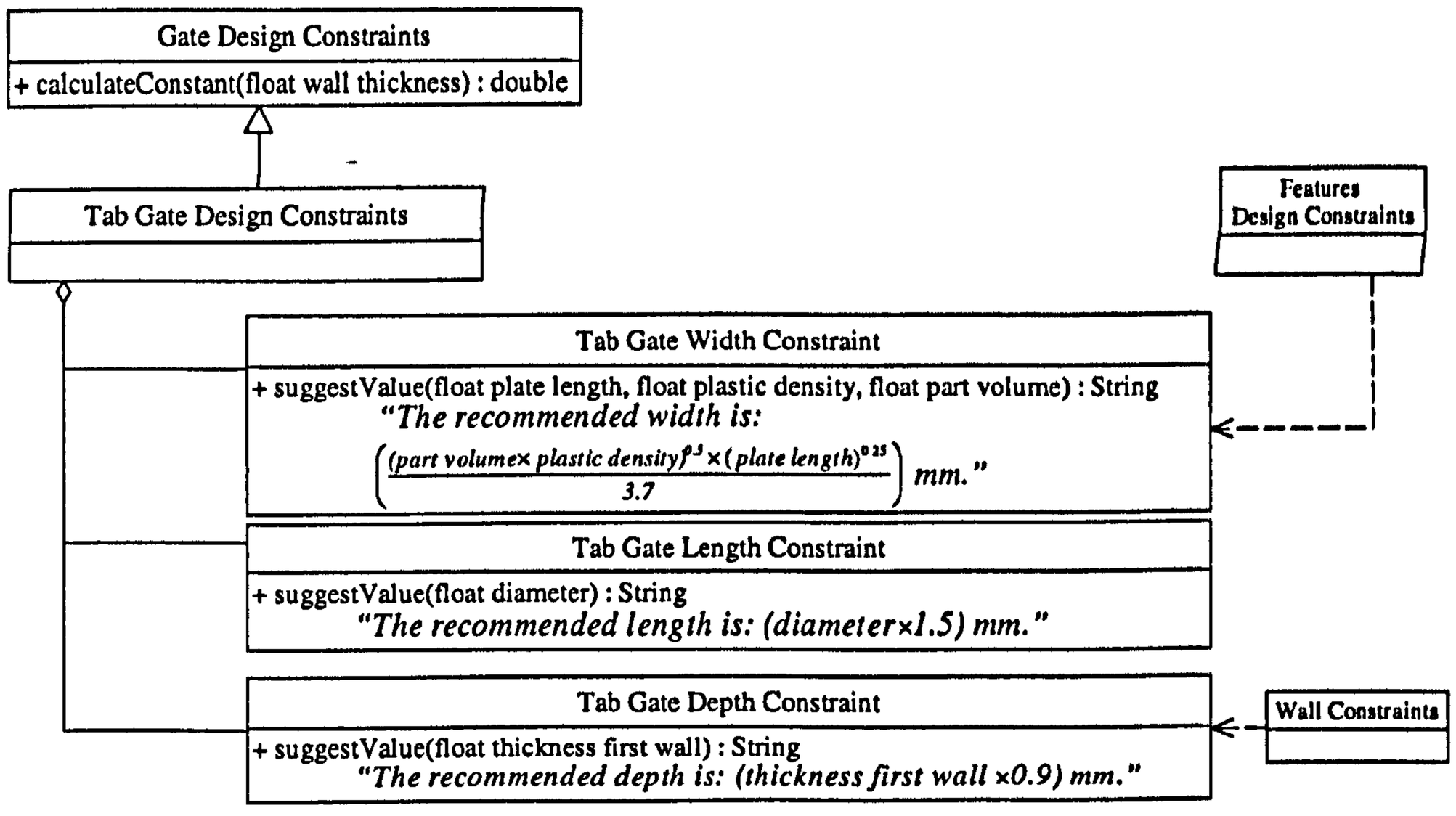


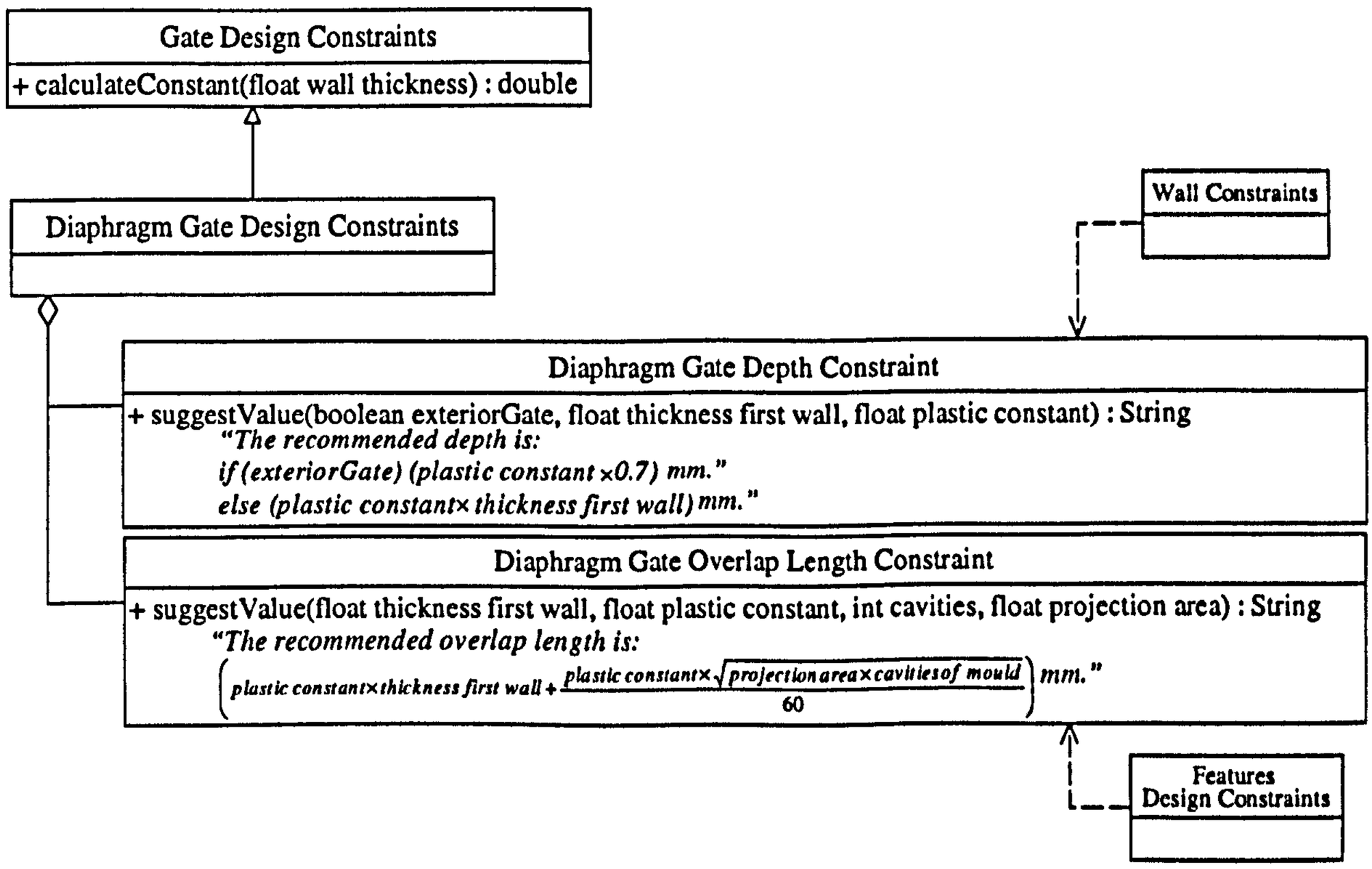
E.5.3.3 Gating system design constraints

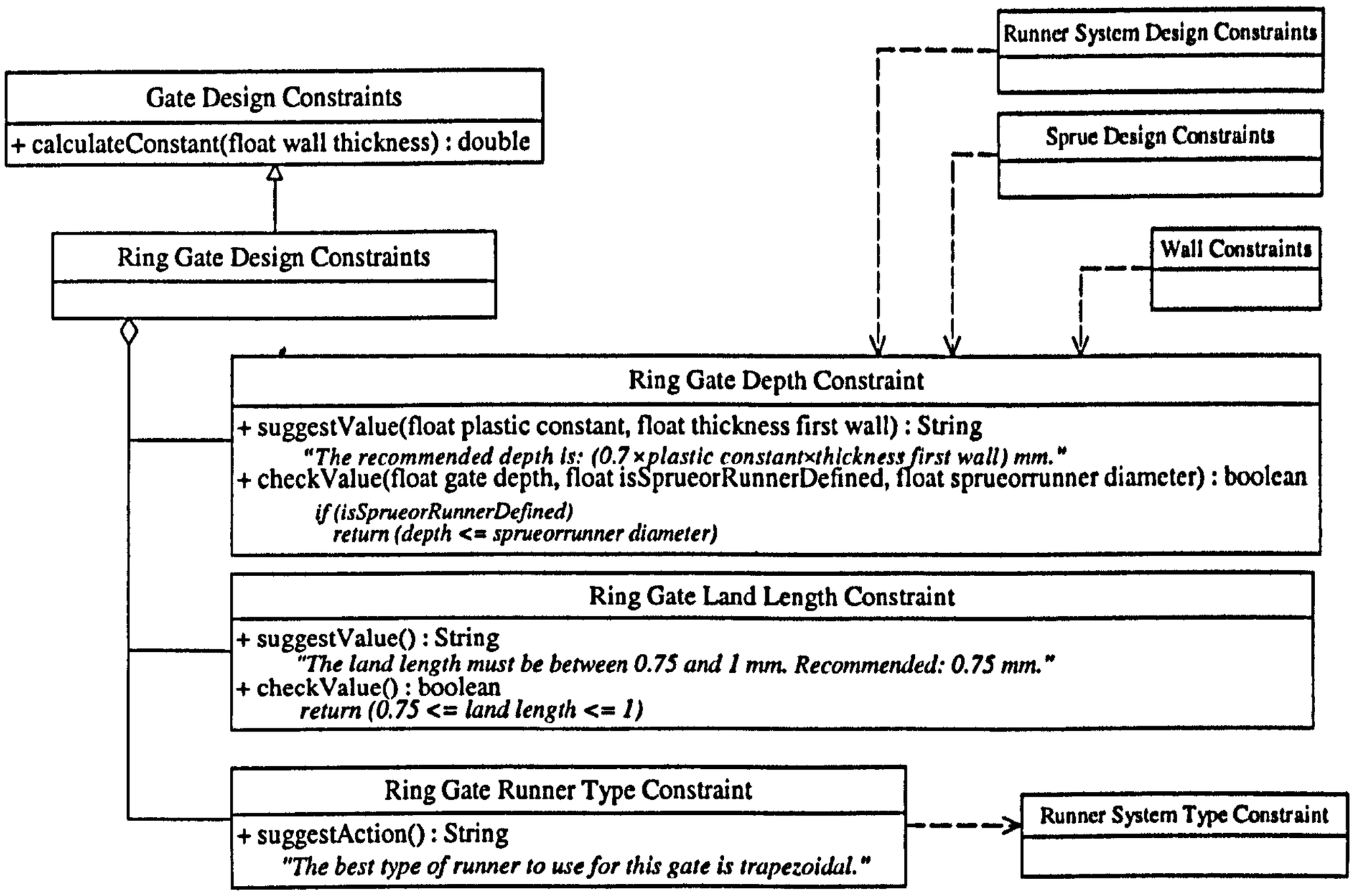


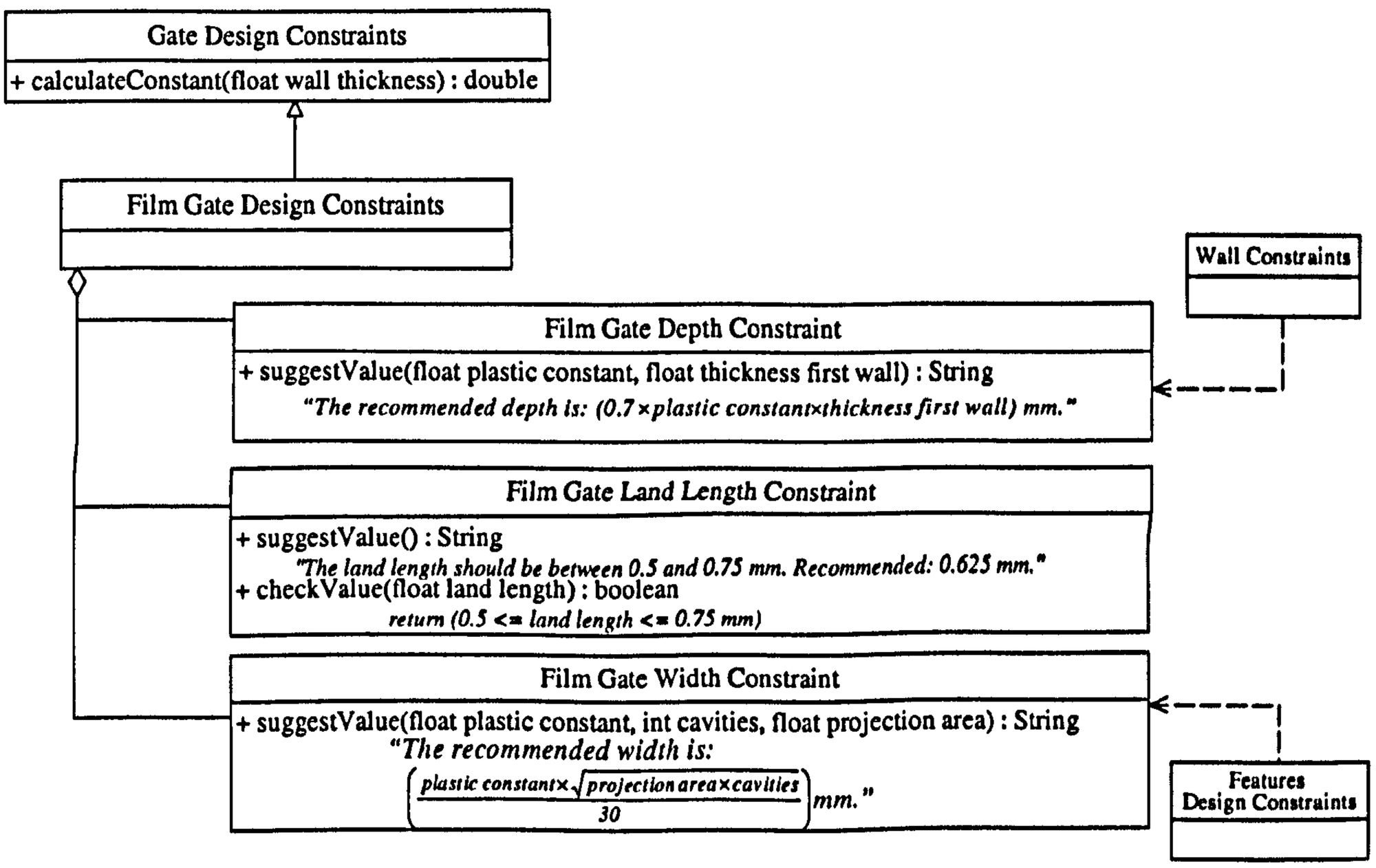

Gate Design Constraints
+ calculateConstant(float wall thickness) : double
Sprue Gate Design Constraints
Sprue Gate Diameter Constraint
+ suggestValue(float thickness first wall) : String
"The recommended diameter is: (thickness of first wall + 1)"
Wall Constraints
Sprue Gate Draft Angle Constraint
+ suggestValue() : String
"The draft angle should be more than 1°"
+ checkValue(float draft angle) : boolean
return (draft angle >= 1)

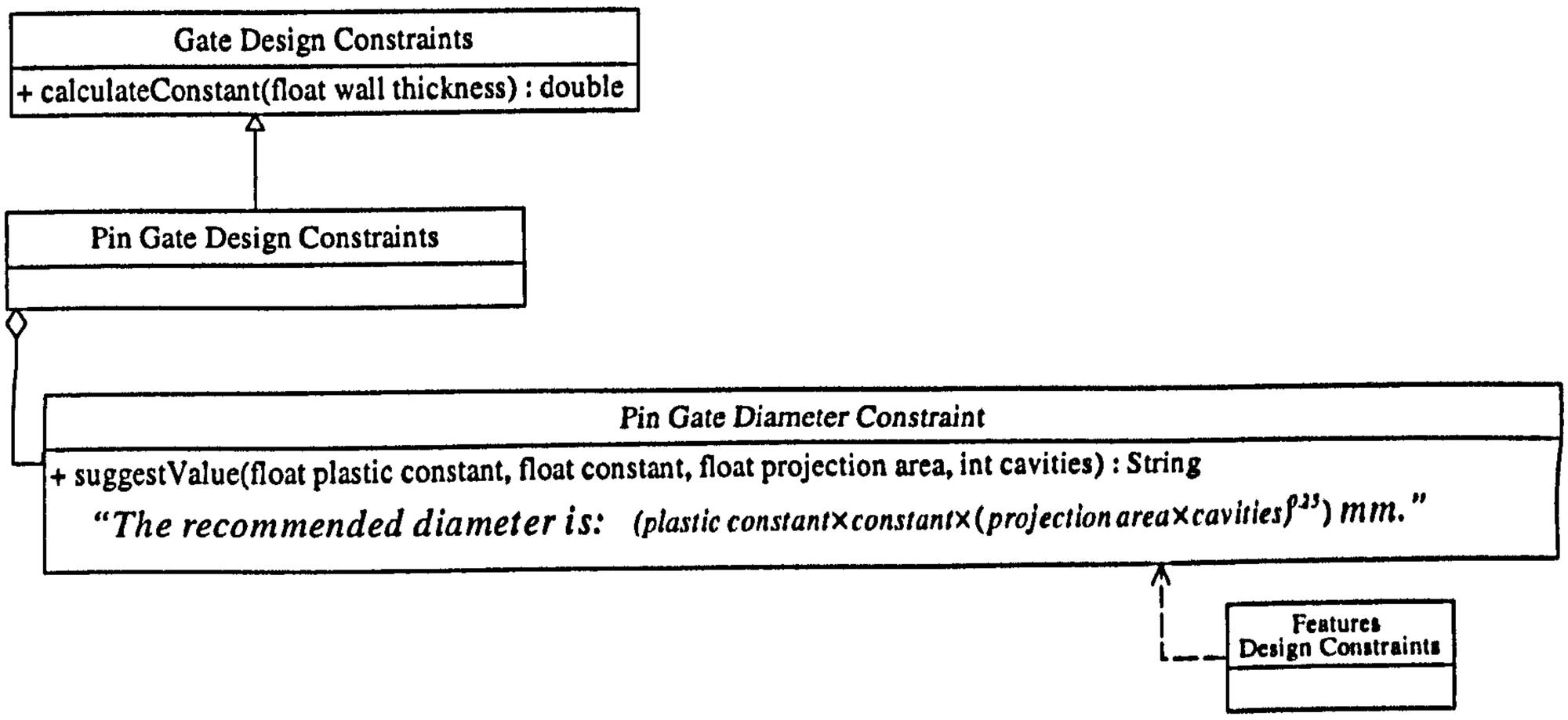

Gate Design Constraints
+ calculateConstant(float wall thickness) : double
Edge Gate Design Constraints
Edge Gate Width Constraint
+ suggestValue(float plastic_constant, int cavities, float projection_area) : double
"The recommended width is:
$\left(\frac{\text{plastic constant} \times \sqrt{\text{projection area} \times \text{cavities}}}{30}\right)$ mm."
Features Design Constraints
Edge Gate Depth Constraint
+ suggestValue(float thickness first wall, float plastic constant) : String
"The recommended depth is: (plastic constant × thickness first wall) mm."
Wall Constraints
Edge Gate Land Length Constraint
+ suggestValue() : String
"The land length should be between 0.5 and 0.75 mm. Recommended: 0.625 mm."
+ checkValue(float land length) : boolean
return (0.5 <= land length <= 0.75 mm)


Gate Design Constraints
+ calculateConstant(float wall thickness) : double
Tab Gate Design Constraints
Features Design Constraints
Tab Gate Width Constraint
+ suggestValue(float plate length, float plastic density, float part volume) : String
"The recommended width is:
$\left(\frac{(\text{part volume} \times \text{plastic density})^{3} \times (\text{plate length})^{0.25}}{3.7}\right)$ mm."
Tab Gate Length Constraint
+ suggestValue(float diameter) : String
"The recommended length is: (diameter×1.5) mm."
Tab Gate Depth Constraint
+ suggestValue(float thickness first wall) : String
"The recommended depth is: (thickness first wall ×0.9) mm."
Wall Constraints

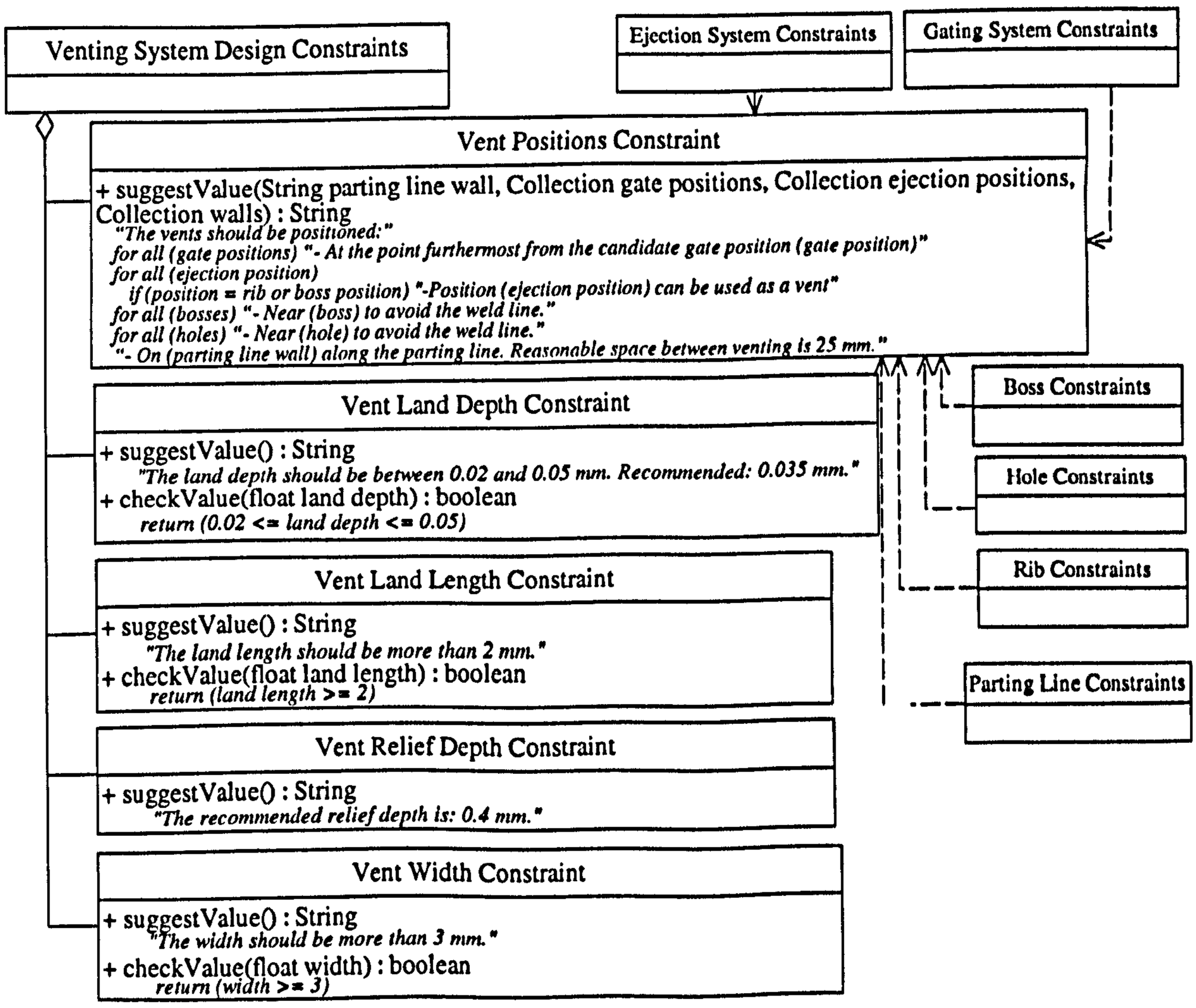

Gate Design Constraints
+ calculateConstant(float wall thickness) : double
Diaphragm Gate Design Constraints
Wall Constraints
Diaphragm Gate Depth Constraint
+ suggestValue(boolean exteriorGate, float thickness first wall, float plastic constant) : String
"The recommended depth is:
if (exteriorGate) (plastic constant ×0.7) mm."
else (plastic constant× thickness first wall) mm."
Diaphragm Gate Overlap Length Constraint
+ suggestValue(float thickness first wall, float plastic constant, int cavities, float projection area) : String
"The recommended overlap length is:
$\left(\text{plastic constant} \times \text{thickness first wall} + \frac{\text{plastic constant} \times \sqrt{\text{projection area} \times \text{cavities of mould}}}{60}\right)$ mm."
Features Design Constraints


Gate Design Constraints
+ calculateConstant(float wall thickness) : double
Ring Gate Design Constraints
Runner System Design Constraints
Sprue Design Constraints
Wall Constraints
Ring Gate Depth Constraint
+ suggestValue(float plastic constant, float thickness first wall) : String
"The recommended depth is: (0.7 × plastic constant × thickness first wall) mm."
+ checkValue(float gate depth, float isSprueorRunnerDefined, float sprueorrunner diameter) : boolean
if (isSprueorRunnerDefined)
return (depth <= sprueorrunner diameter)
Ring Gate Land Length Constraint
+ suggestValue() : String
"The land length must be between 0.75 and 1 mm. Recommended: 0.75 mm."
+ checkValue() : boolean
return (0.75 <= land length <= 1)
Ring Gate Runner Type Constraint
+ suggestAction() : String
"The best type of runner to use for this gate is trapezoidal."
Runner System Type Constraint

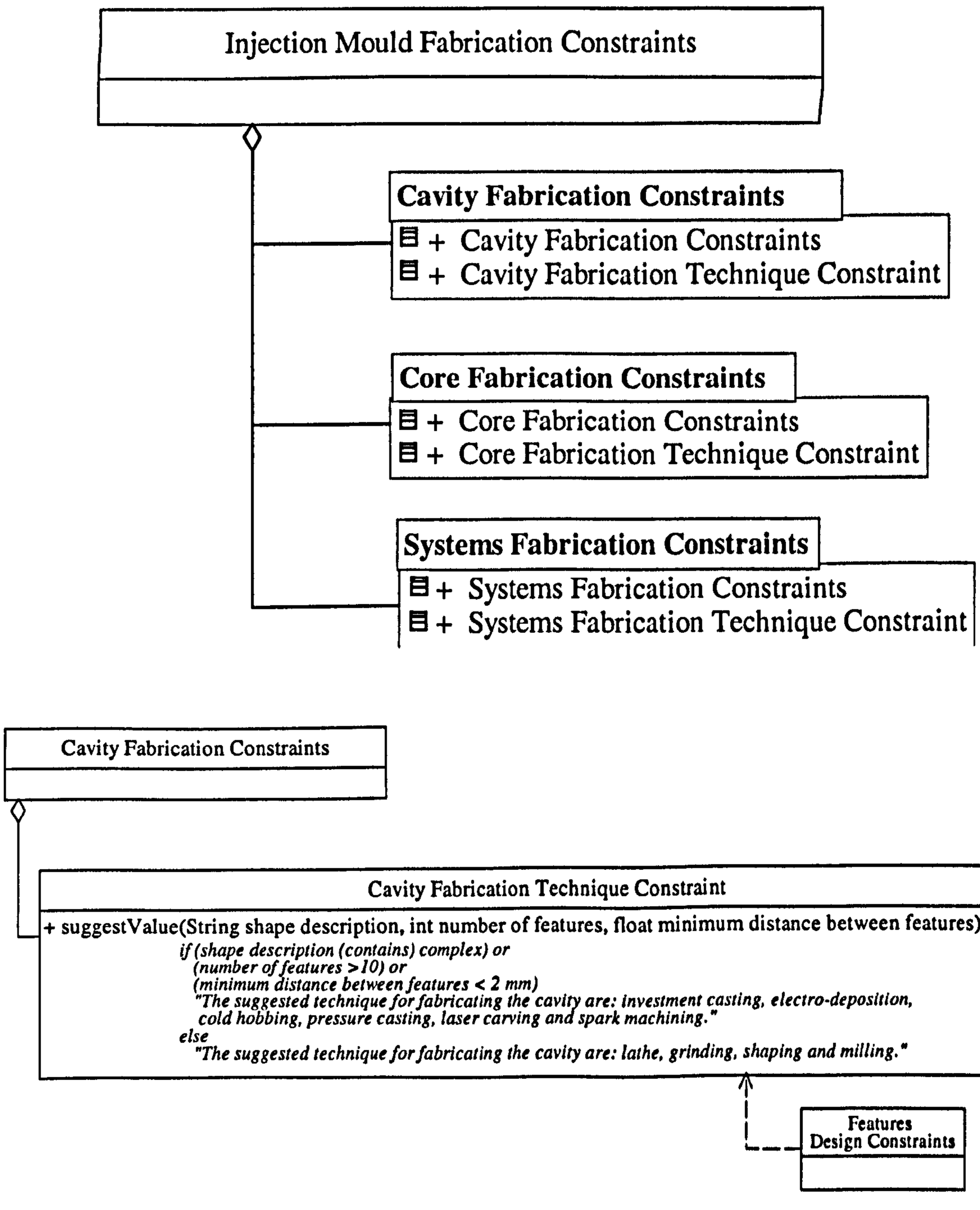

Gate Design Constraints
+ calculateConstant(float wall thickness) : double
Film Gate Design Constraints
Wall Constraints
Film Gate Depth Constraint
+ suggestValue(float plastic constant, float thickness first wall) : String
"The recommended depth is: (0.7 × plastic constant × thickness first wall) mm."
Film Gate Land Length Constraint
+ suggestValue() : String
"The land length should be between 0.5 and 0.75 mm. Recommended: 0.625 mm."
+ checkValue(float land length) : boolean
return (0.5 <= land length <= 0.75 mm)
Film Gate Width Constraint
+ suggestValue(float plastic constant, int cavities, float projection area) : String
"The recommended width is:
(plastic constant × √(projection area × cavities) / 30) mm."
Features Design Constraints

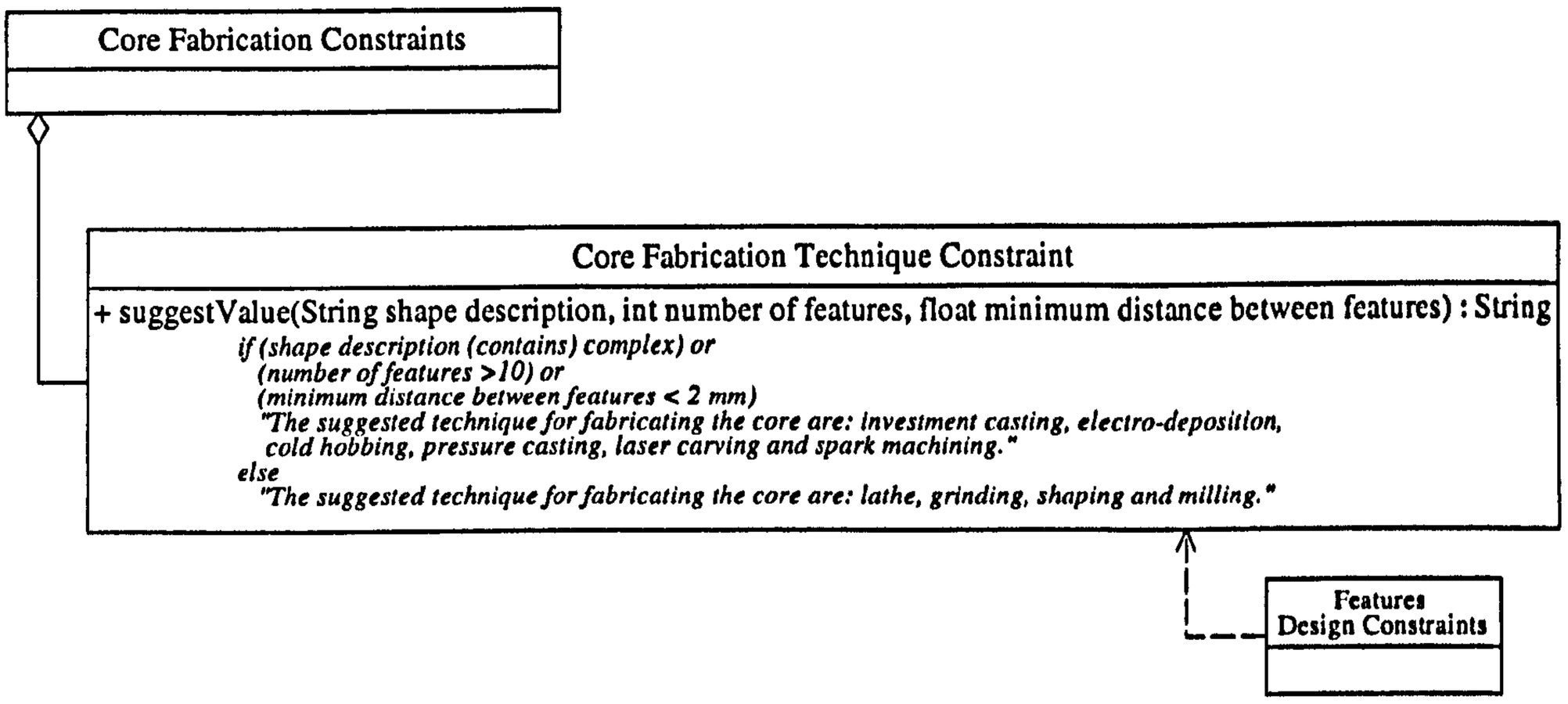


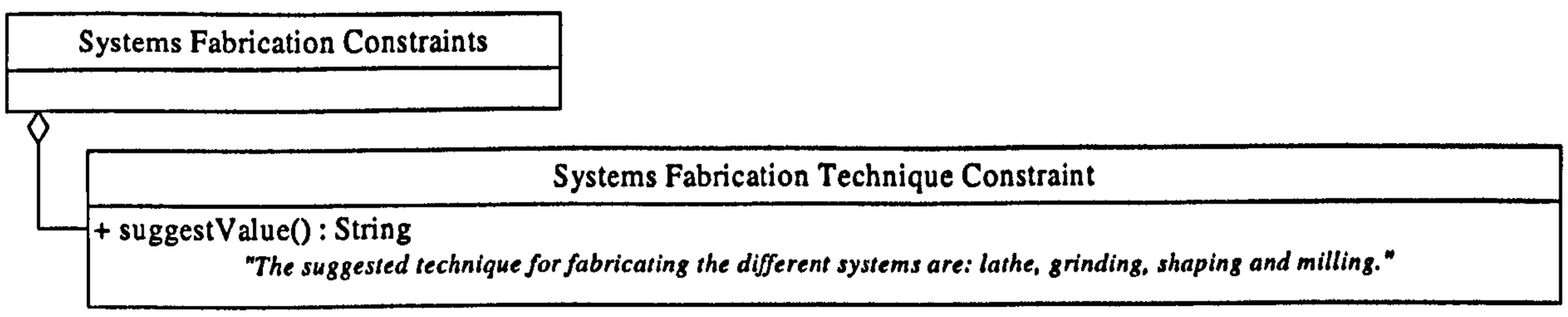
E.5.4 Venting system design constraints



E.6 Mould fabrication constraints




Core Fabrication Constraints
Core Fabrication Technique Constraint
+ suggestValue(String shape description, int number of features, float minimum distance between features) : String
if (shape description (contains) complex) or
(number of features >10) or
(minimum distance between features < 2 mm)
"The suggested technique for fabricating the core are: investment casting, electro-deposition,
cold hobbing, pressure casting, laser carving and spark machining."
else
"The suggested technique for fabricating the core are: lathe, grinding, shaping and milling."
Features
Design Constraints


Systems Fabrication Constraints
Systems Fabrication Technique Constraint
+ suggestValue() : String
"The suggested technique for fabricating the different systems are: lathe, grinding, shaping and milling."

Appendix F

Software User Guide

F.1 Introduction

This appendix describes how the KdCPD system prototype, described in chapter 11, is used.

F.2 Software and hardware requirements

The KdCPD system can be accessed from any PC with the following hardware and software capabilities:

Hardware requirements

- Processor: Pentium 133 mhz or higher
- RAM: 32 MB or higher
- Operative System: Windows 95/98/NT/2000/ME/XP, MacPowerPC or Solaris
- Browser: Microsoft Internet Explorer 5.0 or later/ Netscape 4.6.1 or later.
- Internet connection (LAN or modem connection)

Software requirements

Additional software requires to be installed in the end user's computer before accessing for the first time the KdCPD system. This software is free to use and provides the capability of displaying the 3D geometric model and of providing collaborative tools. In case it has not been previously installed, the software can be downloaded from the following address:

- Java Runtime Environment: http://java.sun.com/j2se/1.4.2/download.html - version 1.4.2 or higher (Sun Microsystems 2003)

- Java3D Runtime Environment: http://java.sun.com/products/java-media/3D/download.html (Sun Microsystems 2003)
- Net Meeting: http://www.microsoft.com/windows/netmeeting/download/default.asp (Microsoft Corporation 1996-2000)
- MSN messenger: http://messenger.msn.com (Microsoft Corporation 1996-2000)

In order to make use of Net Meeting collaborative tools, a .NET Passport account on the MSN network is required. This can be obtained from the msn site (http://login.passport.net)

F.3 Accessing the KdCPD system

After having an Internet connection available, the KdCPD system web site is accessed by opening an Internet browser and typing in the address field the WWW address shown in figure F.1.

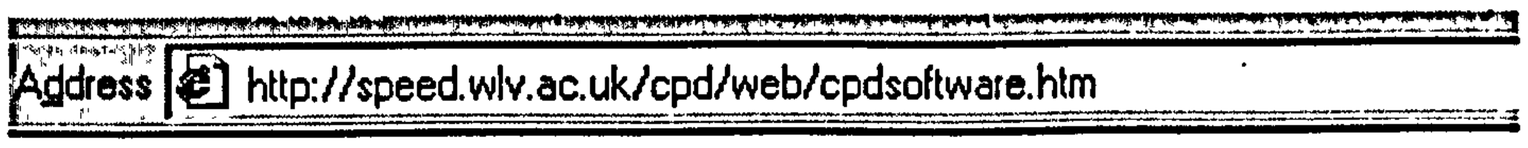


Figure F.1 KdCPD system WWW address

Immediately, the presentation page shown in figure F.2 is loaded into the browser. On this page, the username and password requires to be typed in order to access the KdCPD system. This authentication enables the provision of a customised graphical user interface.

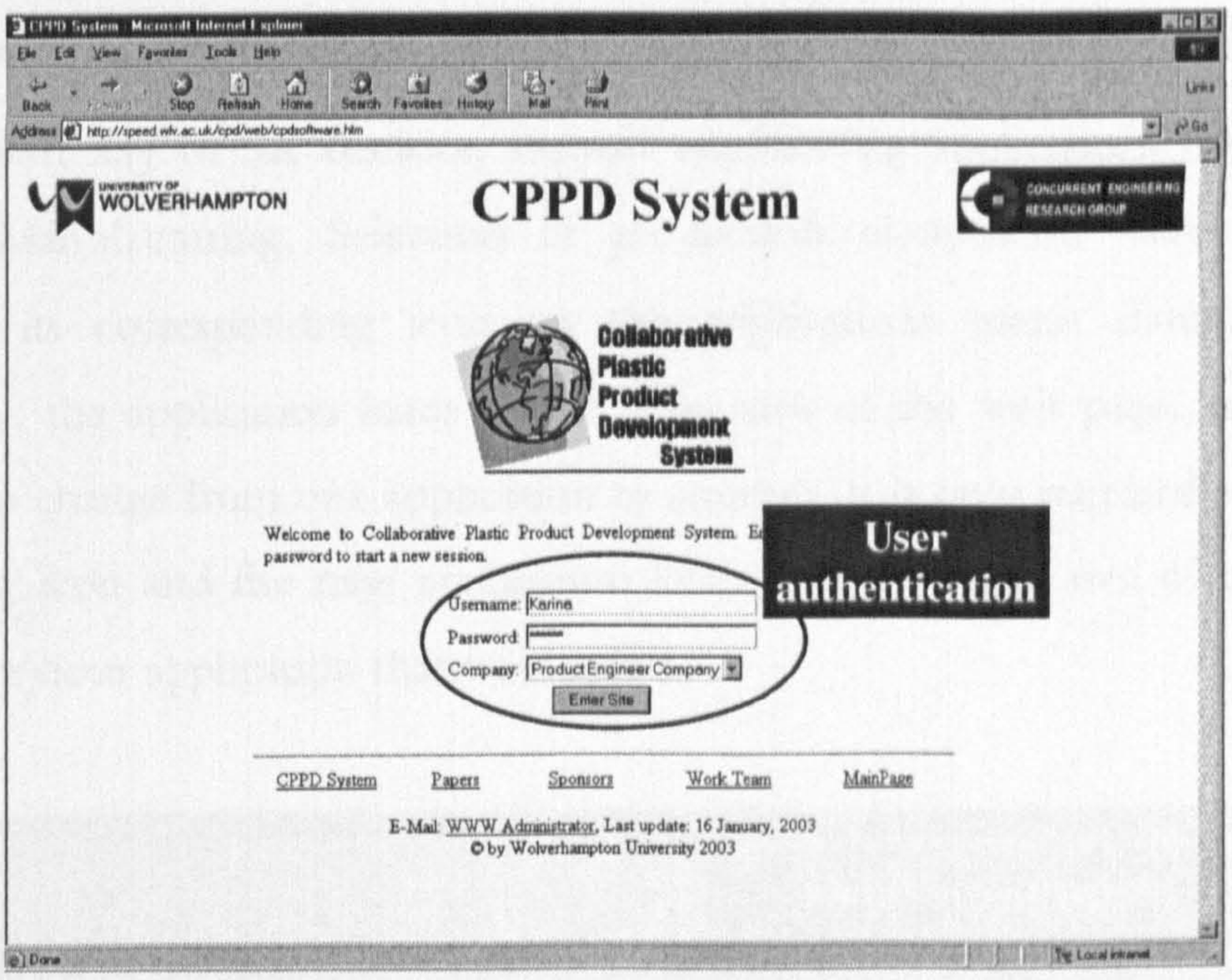


Figure F.2 KdCPD system login web page

After the username and password had been authenticated the KdCPD system main page is loaded into the browser. This page contains a menu of decision support engineering applications to the left of the page, and a main area for loading the applications (see figure F.3).

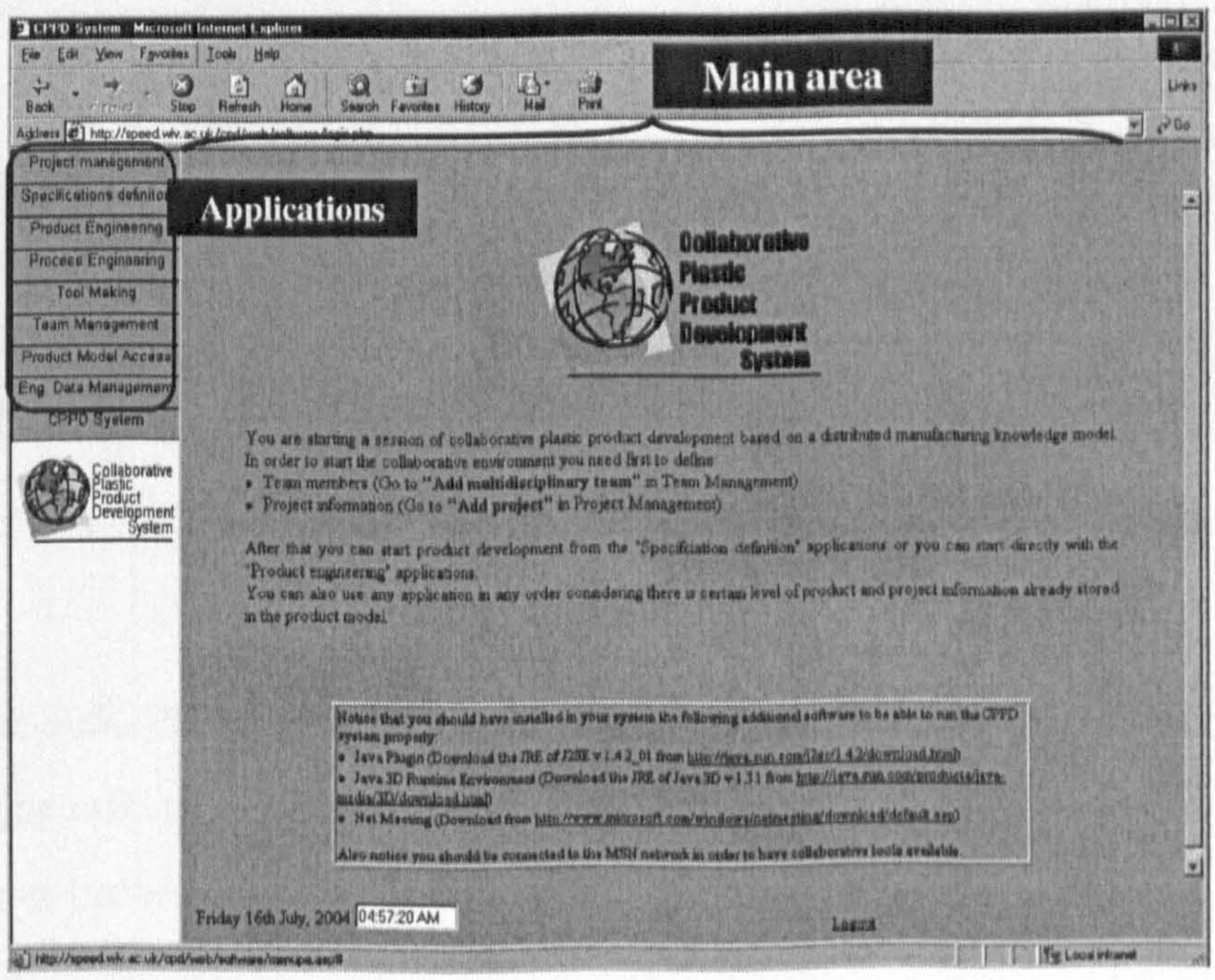


Figure F.3 KdCPD system web site main page

F.3.1 Accessing the decision support engineering applications

In order to start any of the decision support engineering applications (Design Session, Design for Manufacturing, Selection of production equipment, Mould design and Fabrication), its corresponding icon on the applications menu should be clicked. Automatically, the application loads in the main area of the web page, as illustrated in figure F.4. To change from one application to another, it is only required to click on the corresponding icon and the new application loads into the main area of the web page, closing the previous application that was loaded.

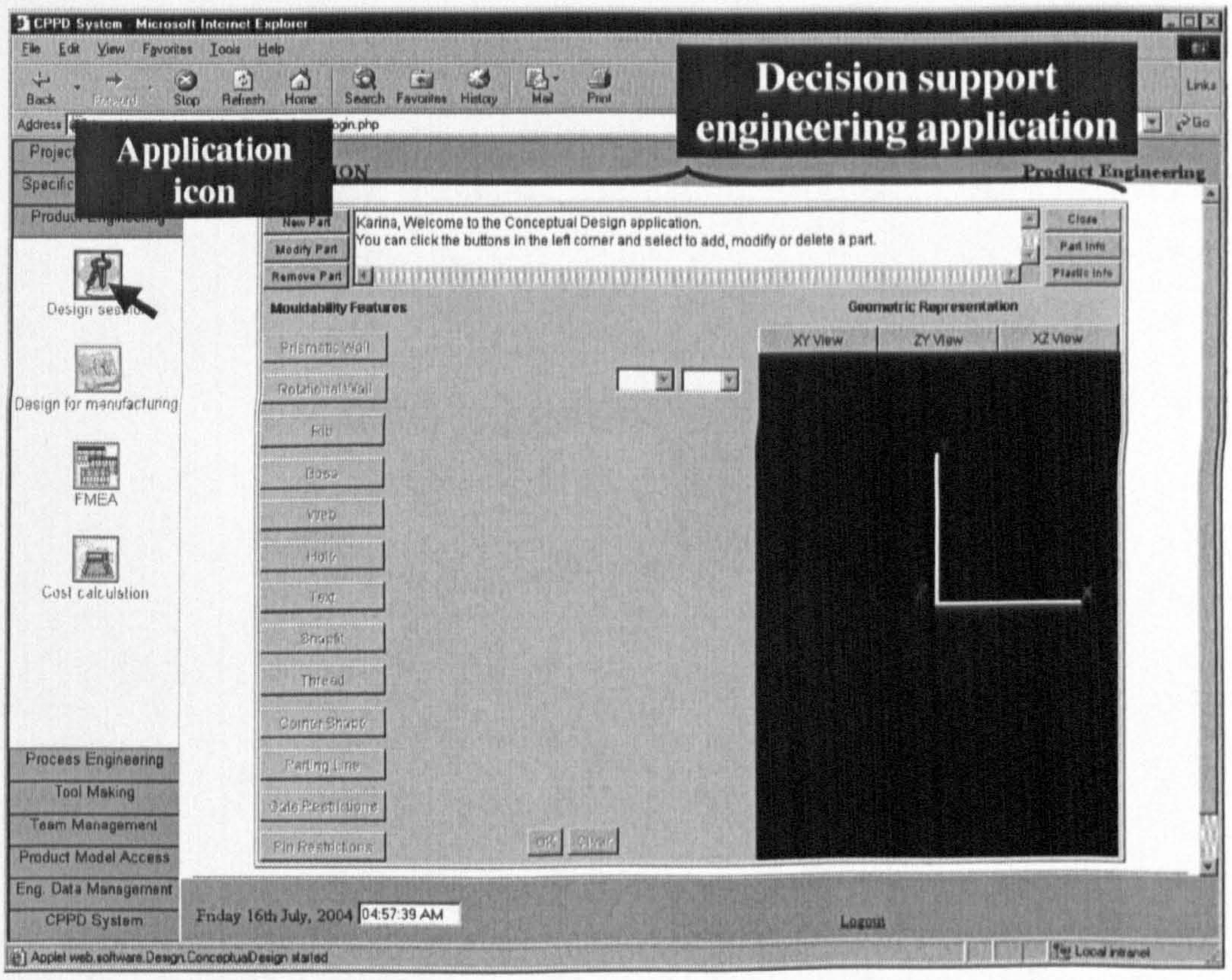


Figure F.4 Starting an engineering application in the KdCPD system

For closing the KdCPD system, the "closing" icon should be clicked. This icon is located in the top right corner of the window. This action will close both the KdCPD system and the Internet browser.

F.3.2 Starting the NetMeeting communication tools

In order to use NetMeeting communication tools along with the KdCPD system, the geographically distributed team members should be running the messenger software and the engineering application simultaneously. This messenger software is started by filling the username and password on the sign in window. Once the user is online, the software provides the capability to show who is online at any time. Any engineer can request at any time to start a collaborative session with any of the other team members online. For this, its name should be right clicked and the "Start NetMeeting" option should be selected (see figure F.5).

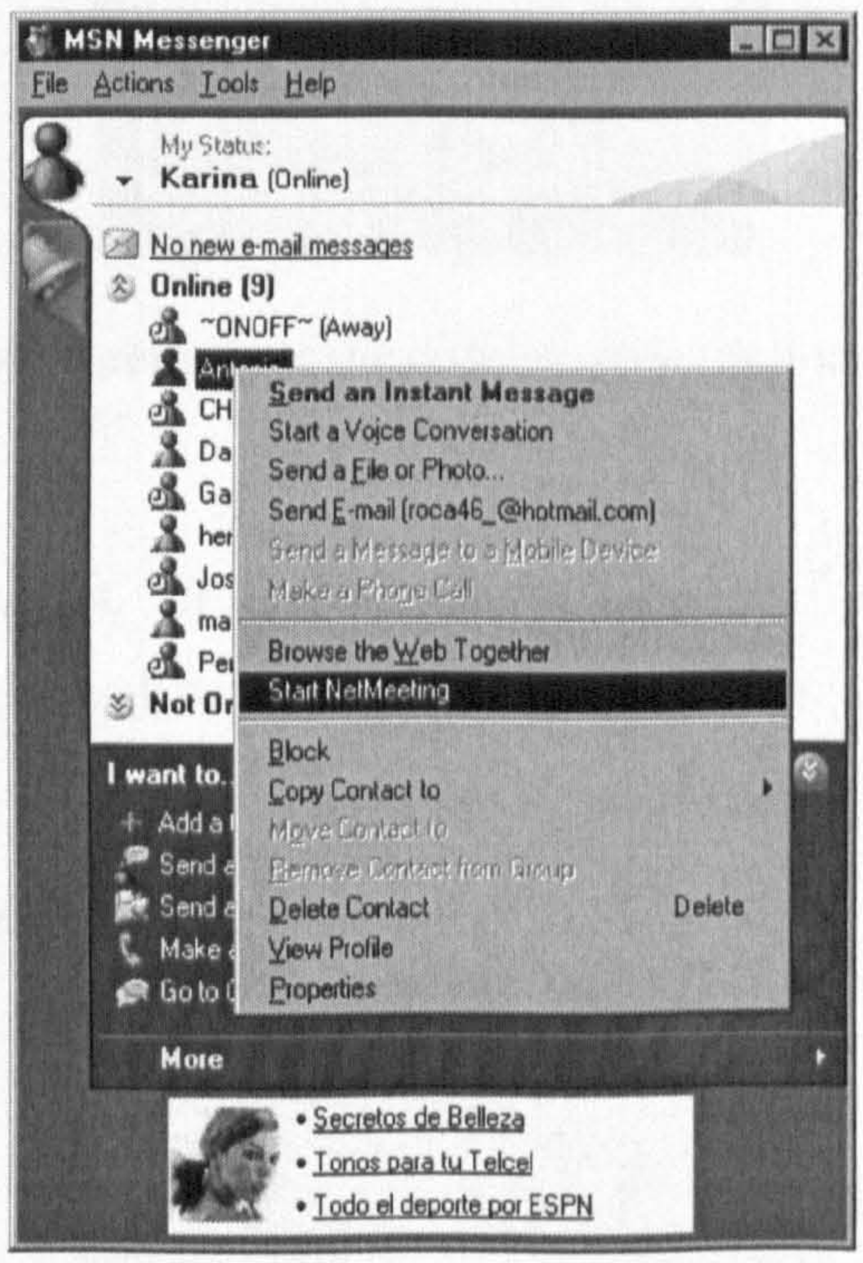


Figure F.5 Starting a collaborative session in MSN Messenger

The software automatically requests the other team member to start a NetMeeting session and he/she is only required to accept in order to load the collaborative environment. This environment consists of a window with the following communication tools: videoconference, sharing applications, chat, whiteboard and sharing files (see figure F.6).

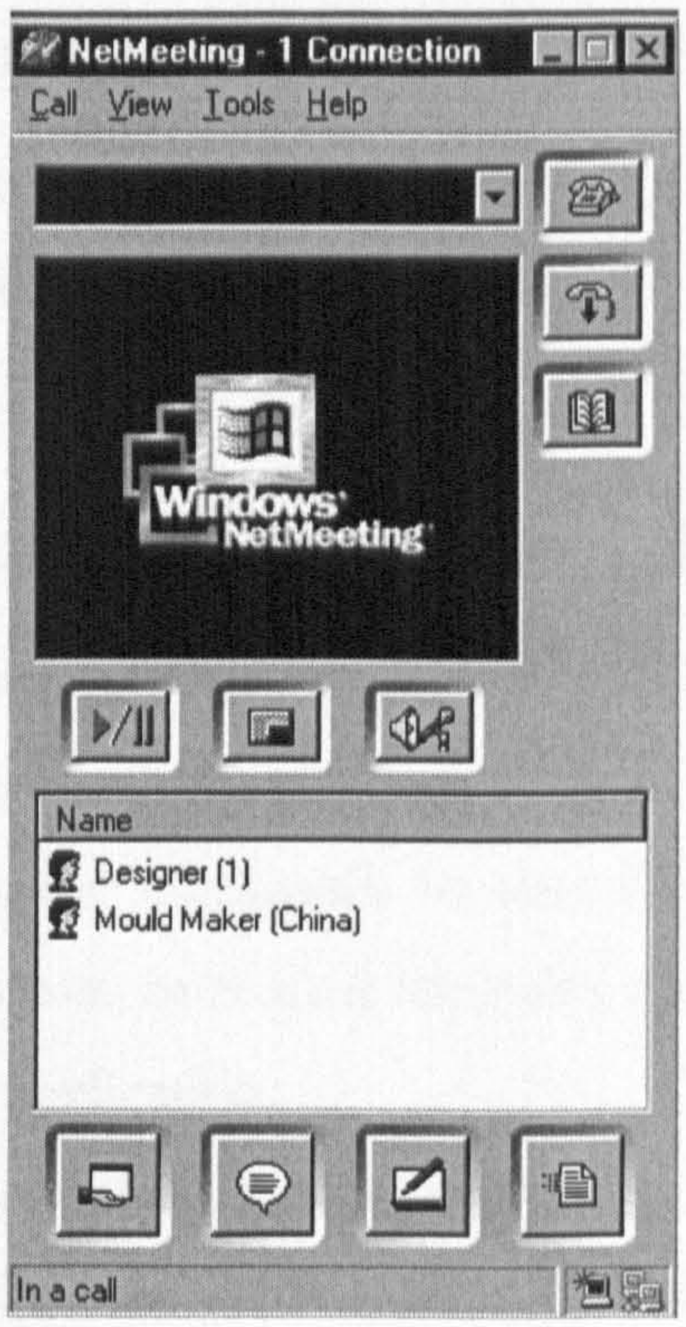


Figure F.6 Graphical user interface of the collaborative tools provided by NetMeeting

In order to share an application, the icon shown in figure F.7 requires to be clicked. This displays the application selected in the computer of both users. The control of the application can then be requested by any of the users. More users can be included in the collaborative session by inviting them to join.

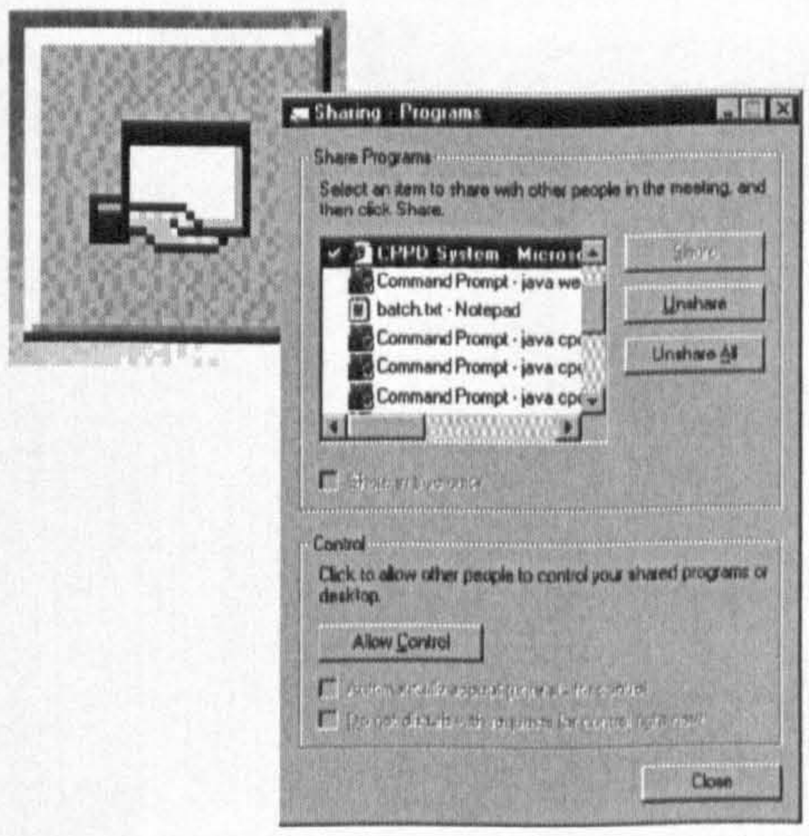


Figure F.7 Requesting to share an application among the geographically distributed team members

F.4 Using the KdCPD system

The usage of each of the implemented decision support engineering application is explained in the following subsections.

F.4.1 Using the Design Session engineering application

Figure F.8 illustrates the design session application graphical user interface, which is composed of the following areas:

- Operation menu: contains the commands to start the definition of a new part, to modify a part, to remove a part, to request the part's definition, to request the plastic's definition and to close the application.
- Feedback area: displays information that guides the user through the design process.
- Features menu: contains the mouldable features, which can be included in a plastic part.
- Input area: displays the required values that define a feature in the Product Model.
- Geometric representation area: displays the 3D geometric model of the part.

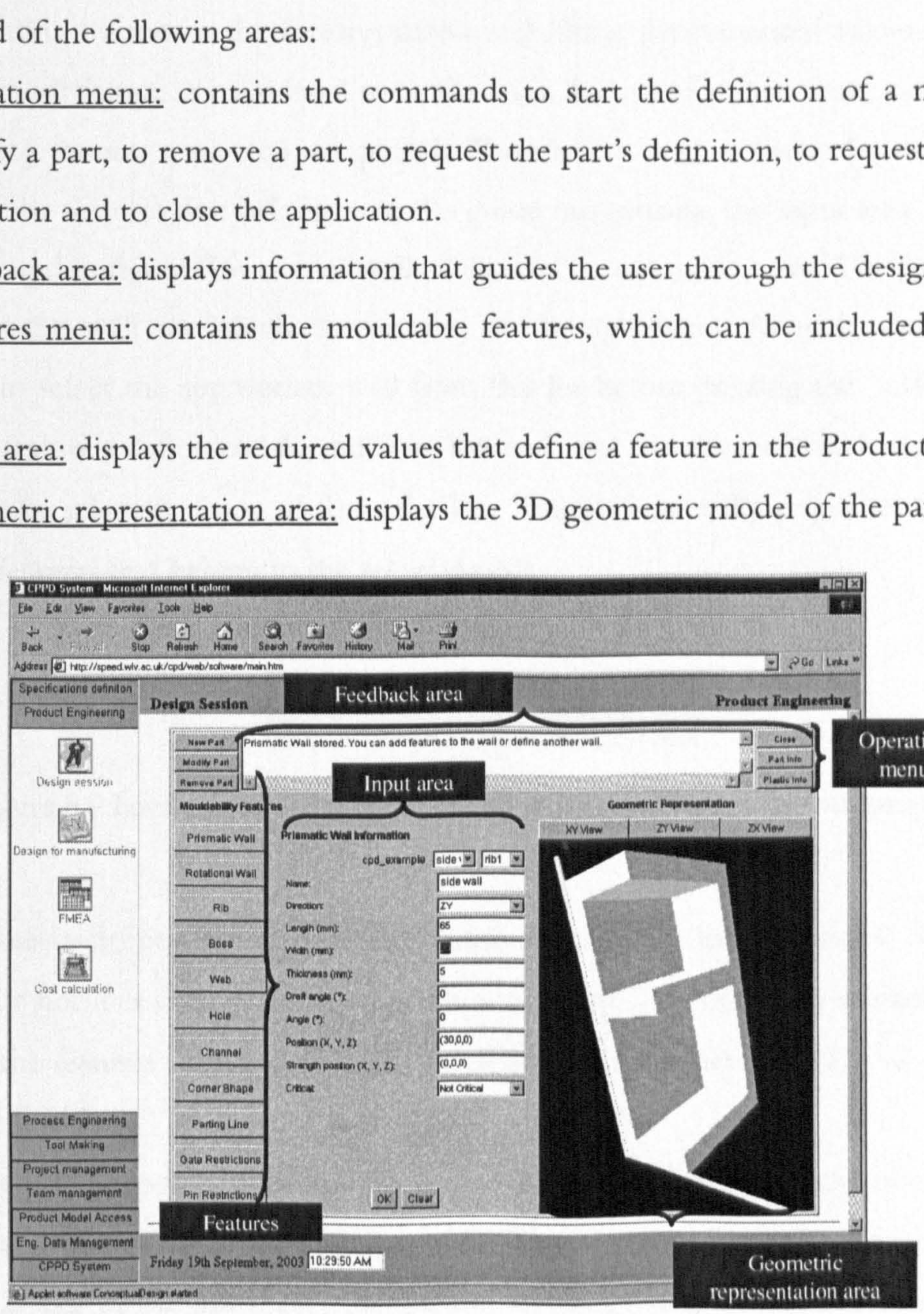


Figure F.8 Graphical user interface of the Design Session application

In order to capture the part definition, the end user needs to click on the "New Part" button in the operation menu. Thereafter, the data of the part, such as name, shape description, length, width, depth and plastic material, needs to be filled in the input area of the application. By pressing the "OK" button the data is captured in the Product Model and the product engineer can start defining the part in terms of features.

The features of the part can be included in the part definition by clicking on its corresponding button in the features menu and filling the requested values in the input area. The wall feature is considered to be the main feature of a plastic part, on which other features (e.g. bosses, text, etc.) are placed. Therefore, a wall needs to be defined before including the features that belong to it. To guide this process, the input area contains two lists illustrated in figure F.9: one of walls, followed by one of features. The first list is used to specify the wall on which the features are being placed. As such, the end user is required to select the appropriate wall from this list before pressing the "OK" button to store the feature's definition. In addition, the end user can review which features belong to a wall, by selecting the wall from the list. This automatically populates the other list with the features that belong to the selected wall.

Figure F.9 List of walls and features defined during the Design Session application

Each feature is defined by its name and attributes, such as length, width, thickness and whether or not it is critical for the part's functionality. The criticality attribute is used to prioritise the features of the part during the design for manufacturing (DFM) analysis.

For orientation purposes, every feature is located in a 3D direction plane of the three axis coordinate system (see figure F.10). It is assumed that the opening direction of the mould is in the Z axis of the 3D space. Hence, the features that are facing the positive Z axis would be imprinted by the core of the mould. On the other hand, the features that are facing the negative Z axis would be imprinted by the cavity of the mould.

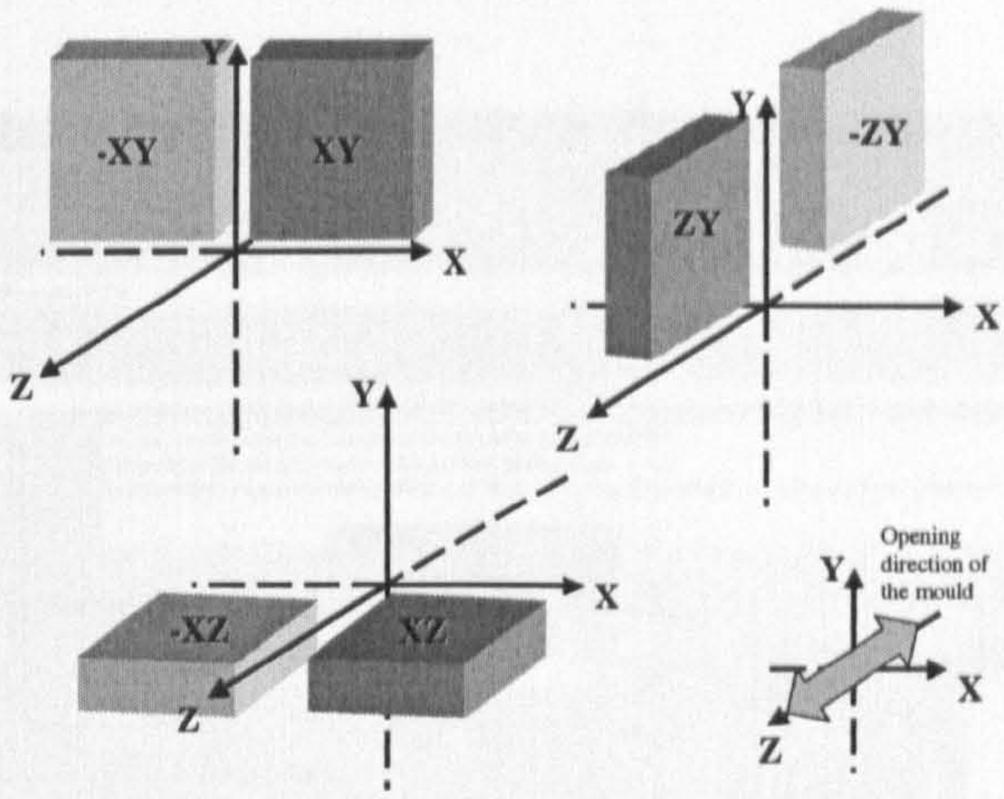


Figure F.10 Direction planes for orienting features in the 3D space

The feature's definition is confirmed by pressing the "OK" button as shown in figure F.8. Following this, a message is displayed in the feedback area to confirm the successful capturing of the data or any problems. At the same time, the 3D virtual geometric model of the feature is displayed in the geometric representation area (see figure F.8).

Finally, the end user could consult the part's data that has been defined by clicking the "Part Info" button in the operation menu.

F.4.2 Using the Design for Manufacturing engineering application

Figure F.11 illustrates the design for manufacturing application graphical user interface, which is composed of the following areas:

- Operation menu: contains the commands to load a part, to start the design for manufacturability analysis, to check the feedback log file, to request the part's definition, to request the plastic's definition and to close the application.
- Feedback area: displays information that guides the user through the design for manufacturing analysis.
- Features menu: contains the mouldable features, which can be included in a plastic part.
- Input area: displays the values that define the feature.

- Geometric representation area: displays the 3D geometric model of the part.

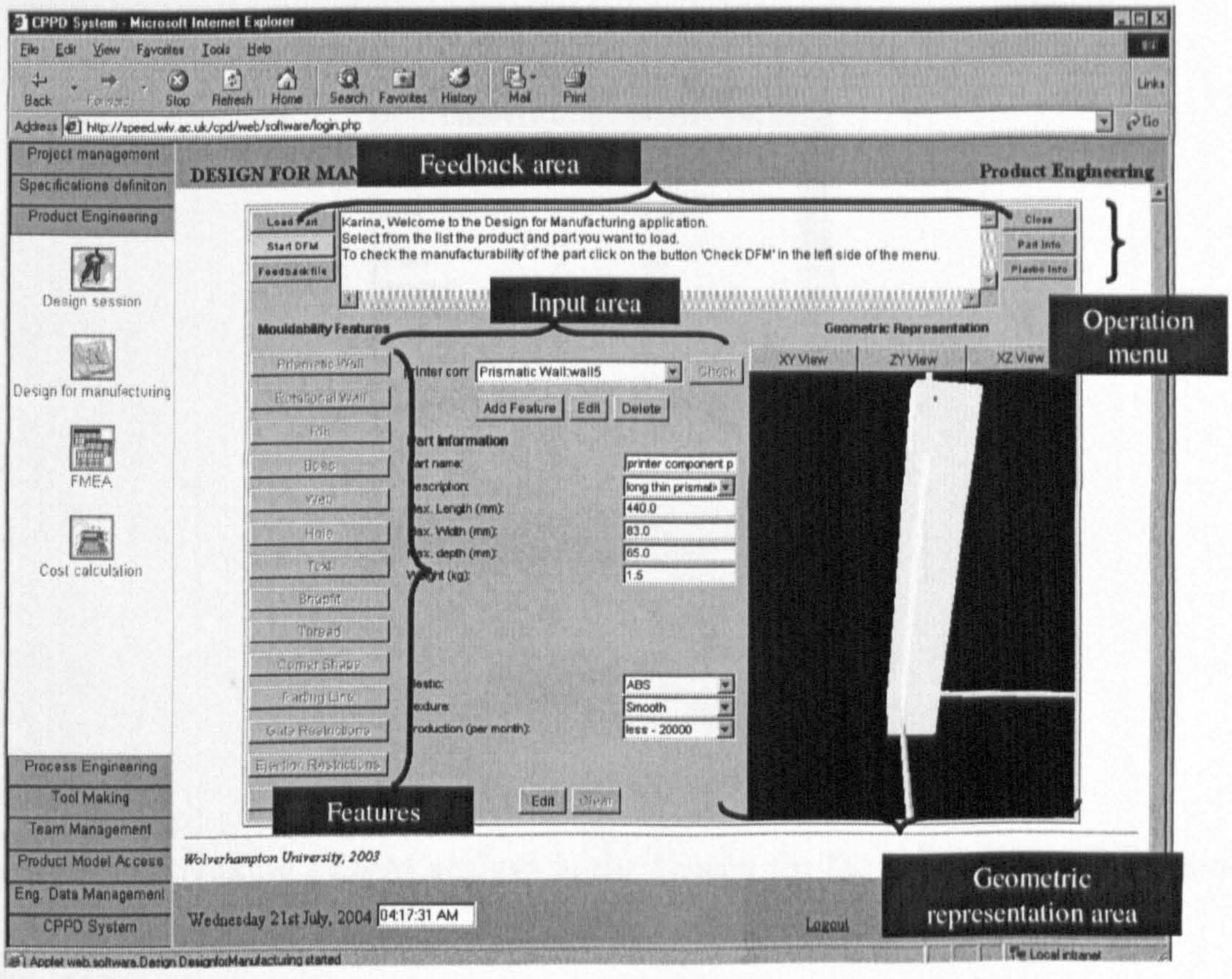


Figure F.11 Graphical user interface of the Design for Manufacturing application

To start this application, the end user first needs to load a part into the application. This is done by clicking on the "Load Part" button in the operation menu. Following this, it is required to select the name of the part from a list of parts and click the "OK" button. The feedback area then displays a text informing the user that the part data has been loaded and the analysis can be started by clicking the "Start DFM" button of the operation menu.

By clicking the "Start DFM" button, an extra window appears as illustrated in figure F.12. This window illustrates the features of the part, their criticality and if its manufacturability has already been checked. In addition, the window includes two buttons, which presents the options given to the end user to perform the design for manufacturing analysis:

- To check all the features at once
- To check one individual feature at a time

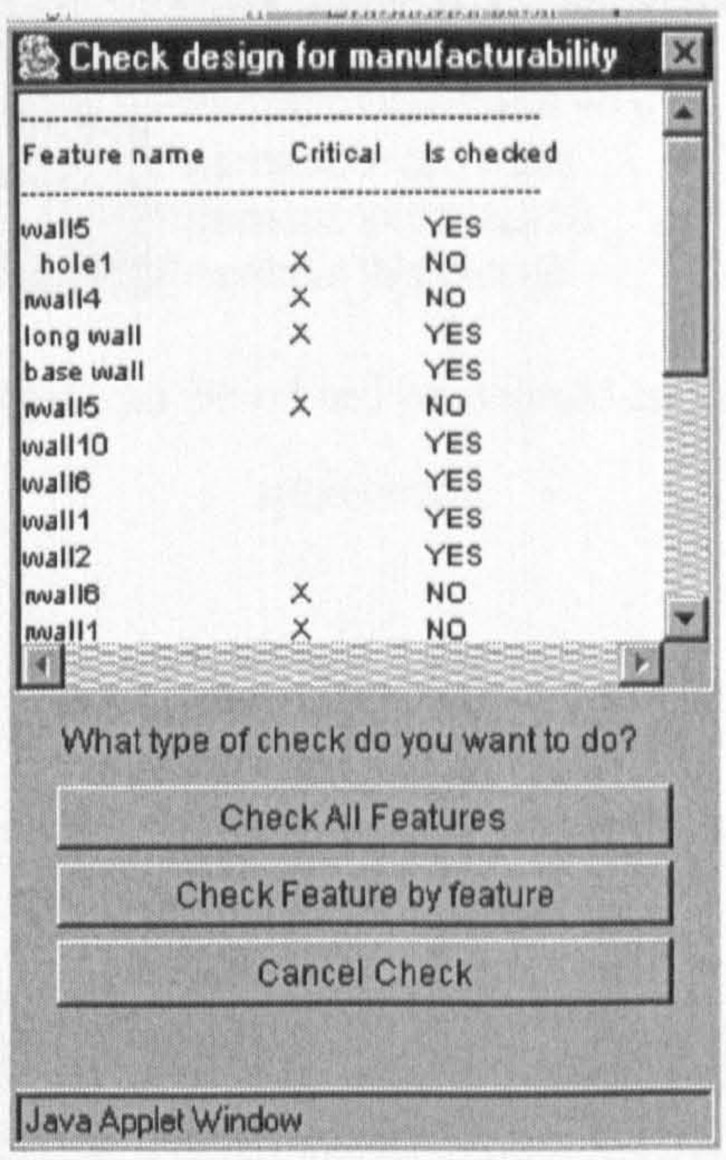


Figure F.12 Invoking a DFM analysis in the Design for Manufacturing application

The feedback advice generated by the design for manufacturing analysis is then displayed in another window and stored in the feedback log file.

In case the part is not within manufacturing constraints, the end user can make use of this application as a design session to add, edit or delete features. As such, the features list shown in figure F.13 presents the complete set of features that was defined for the plastic part.

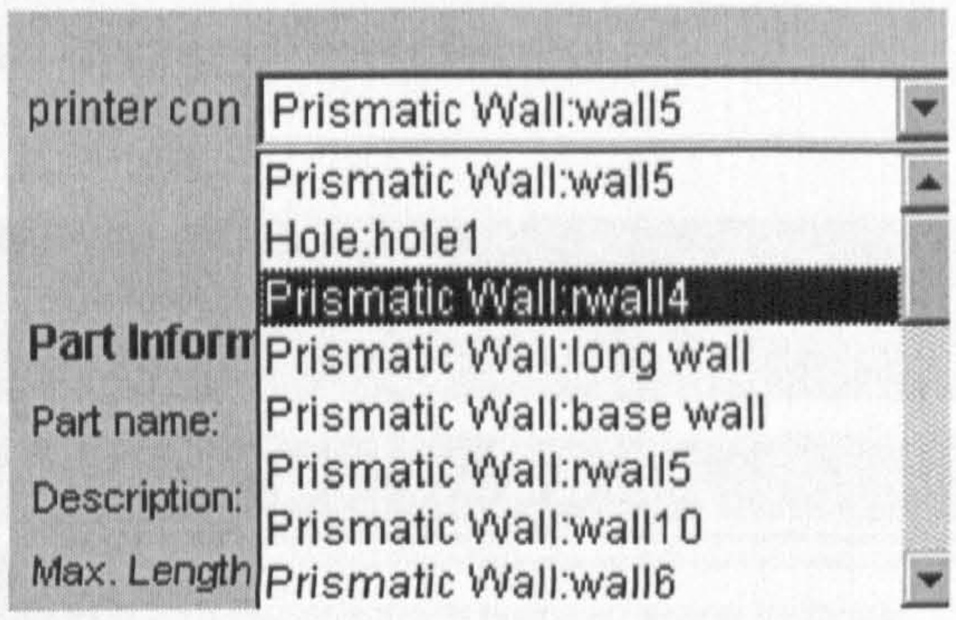


Figure F.13 List of features that can be edited or deleted in the Design for Manufacturing application

In order to edit or delete any of the features, this must first be selected from the list. Then, the action that is preferred to be taken is selected by clicking any of the buttons shown in figure F.14.

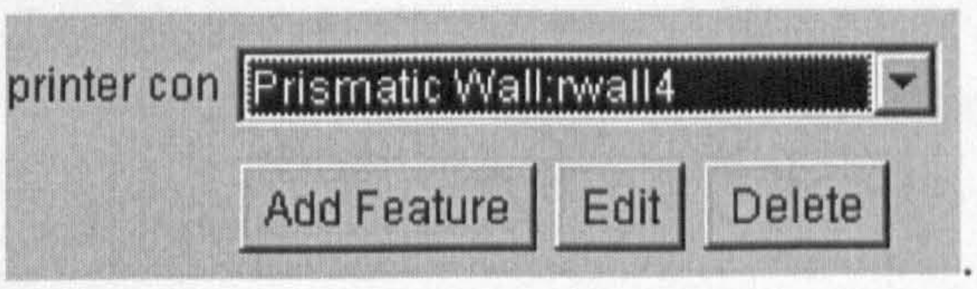


Figure F.14 Add, edit and delete button in the Design for Manufacturing application

The design for manufacturing analysis can be repeated any number of times by following the same process until the part is within manufacturing constraints.

F.4.3 Using the Selection of Production Equipment application

The areas included in the graphical user interface of the selection of production equipment application, shown in figure F.15, are:

- Operation menu: contains the commands to load a part, to select a suitable injection machine for the part, to review the suitable machine characteristics to produce a part, to request the part's definition, to request the plastic's definition and to close the application.
- Feedback area: displays information that guides the user through the selection of a suitable injection machine.
- Machines list: displays a list of machines, which belong to the manufacturer.

- Input area: displays the machine characteristics.

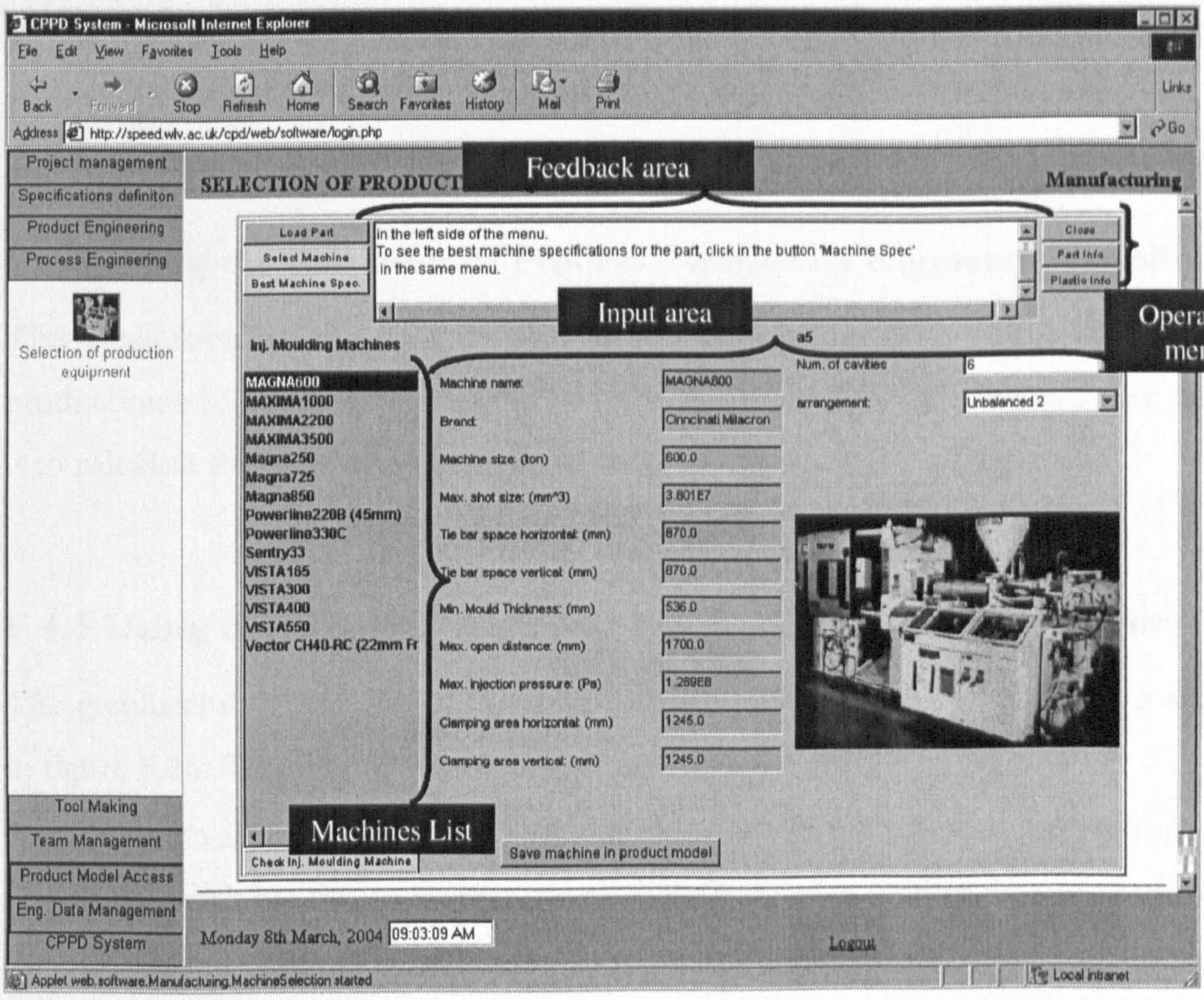


Figure F.15 Graphical user interface of the Selection of Production Equipment application

To start this application, the data of the part needs to be loaded by clicking the "Load Part" button on the operation menu and by selecting the part. Thereafter, the number of cavities and its layout needs to be selected from the list in the input area. In case the mould has already been designed, the number of cavities and its layout are automatically selected. After specifying this data, two actions can be taken:

1. Selecting an injection machine for the plastic part: by clicking its corresponding icon in the operation menu. The machine name and specifications are automatically be displayed in the input area.
2. Reviewing the required machine characteristics: by clicking on the "Best Machine Spec" button in the operation menu. These characteristics are automatically displayed in the input area.

Once the machine that is going to be used has been selected, this can be stored in the Product Model by clicking the "Save Machine in Product Model" button in the input area.

F.4.4 Using the Selection of Process Parameters engineering application

This application has the same interface and is used in the same way as the selection of production equipment application. The only difference is that the functionality requested is to calculate the suitable process parameters.

F.4.5 Using the Mould Design and Fabrication engineering application

The graphical user interface of the mould design and fabrication application is illustrated in figure F.16. This interface contains the following areas:

- Operation menu: contains the commands to load a part, to load a mould, to review the feedback log file, to request the part's definition, to request the plastic's definition, to request the mould definition and to close the application.
- Feedback area: displays information that guides the user through the design of an injection mould.
- Menu of mould components: contains the components, which compose and injection mould.
- Input area: displays the values that define a mould's component in the Product Model.

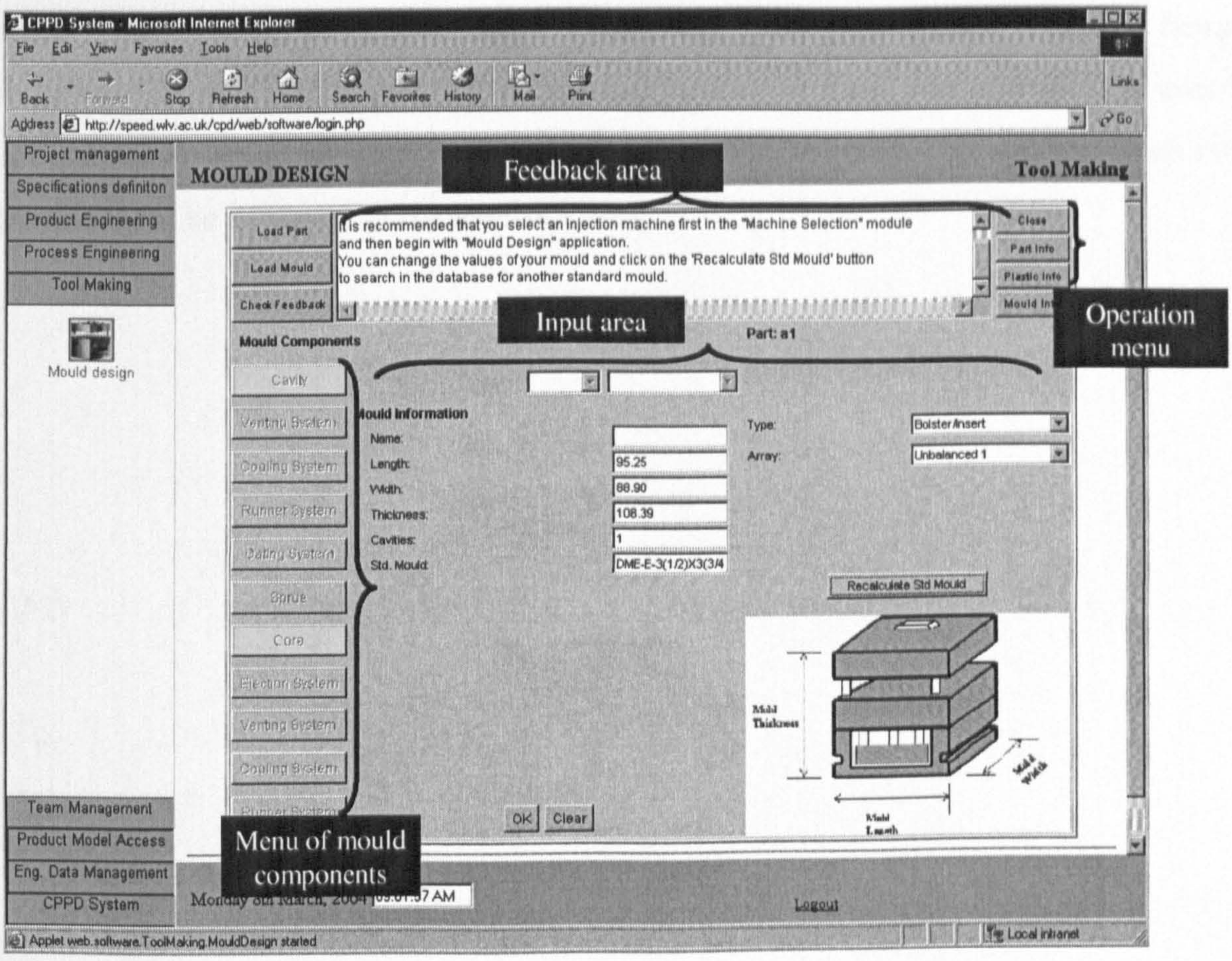


Figure F.16 Graphical user interface of the Mould Design and Fabrication application

In order to start defining the components of the mould, the part needs to be loaded into the application by clicking the "Load Part" button. Following this, the application displays in the feedback area the need to start defining the mould plates before defining other components of the mould. For this, the application displays suitable plate dimensions in the input area. In case the injection machine has already been selected, the number of cavities and their layout are displayed automatically. Otherwise, this data needs to be specified. In case the number of cavities is changed, the dimensions displayed by the application could be recalculated by pressing the "Recalculate Std Mould" button. Once the data of the mould plates has been defined in the input area, the button "OK" needs to be clicked to store this data in the Product Model.

Following this, the components of the mould can be defined by clicking their corresponding icon in the menu of mould components. Automatically, the application

displays feedback advice in the feedback area regarding the component that is being designed. The requested values should then be filled in the input area and they are stored in the Product Model after the end user presses the "OK" button. Any problem with the design would be displayed in the feedback area.